Back to the Roots: Linking User Stories to Requirements Elicitation Conversations

Tjerk Spijkman
fizor. and Utrecht University
Utrecht, the Netherlands
Email: tjerk@fizor.io

Fabiano Dalpiaz
Utrecht University
Utrecht, the Netherlands
Email: f.dalpiaz@uu.nl

Sjaak Brinkkemper
Utrecht University
Utrecht, the Netherlands
Email: s.brinkkemper@uu.nl

***Abstract*—Pre-requirements specification (pre-RS) traceability focuses on tracing requirements back to their sources. In comparison with post-RS traceability, pre-RS traceability is an under-explored area of research. Likely reasons for the limited studies are the scarcity of pre-RS resources, e.g., recorded requirements elicitation conversations such as interviews or workshops, and the challenges of linking requirements to informal, unstructured text. Building on the increasing use of digital communication tools that allow the recording and transcription of conversations, we explore the opportunity of linking requirements to the transcript of a requirements elicitation conversation. We introduce TRACE2CONV, a prototype tool that aims at tracing user story requirements back to the relevant speaker turns in a conversation. TRACE2CONV makes use of NLP techniques to determine the relevant speaker turns. As an initial validation, we take automatically generated transcripts from real-world requirements conversations, and we assess the effectiveness of TRACE2CONV in supporting the process of identifying additional context for the requirements. The validation serves as a formative evaluation that guides the evolution of TRACE2CONV and as a inspiration for future research in the field of *conversational RE*.**

***Keywords*—Requirements Elicitation, User Stories, Natural Language Processing, Conversational RE.**

I. INTRODUCTION

Requirements traceability (RT) refers to the ability to describe and follow the life of a requirement, both forward and backward [1]. Conducting RT is important to identify the sources of a requirement [2], to analyze the impact of a requirement on software engineering artifacts such as code and test cases [3], and to determine dependencies between requirements, also known as horizontal traceability [4].

Depending on whether we look backward or forward from a requirements specification document, we can distinguish between [1]: *Pre-RS* traceability, referring to linking the requirements in a specification to the sources that justify their existence; and *Post-RS* traceability, concerned with the life cycle of a requirement after its inclusion in the specification.

Although the high(er) potential of pre-RS traceability has been recognized already in the 1990s by Gotel [1], Pre-RS traceability is a significantly less explored area of research than post-RS traceability [2]. We agree with Krause *et al.* [2] and argue that this is due to the easier availability of and accessibility to the artifacts. For example, code and test cases (post-RS) are much easier to access by researchers than interview recordings, whiteboard contents, etc. (pre-RS).

The increasing use of digital communication tools (e.g., video-conferencing software with recording and automated captioning), also accelerated by the increased remote work and collaboration that resulted from the COVID-19 pandemic, creates an opportunity to revitalize the research in this area.

We present ongoing research that is part of *conversational RE*: the analysis of requirements elicitation conversations (in short form, requirements conversations) aimed at identifying and extracting requirements-relevant information. Conversational RE sets requirements elicitation conversations as central RE artifacts, in contrast with traditionally studied artifacts such as requirements specification documents [5].

We focus on how to use the verbal communication that is so important in real-world projects but that is largely overlooked in RE research. A few exceptions are the analysis of interview recordings in projects regarding information systems [6], [7] and the analysis of simulated interviews in RE education [8], [9]. Yet, we are not aware of any studies that trace requirements back to requirements conversations.

In this paper, we design and report on TRACE2CONV, a prototype tool that aims at automatically tracing user story requirements [10], [11] back to the segments of a requirements interview that are likely to justify that requirement. In particular, we make the following contributions to the RE field:

- We describe TRACE2CONV and the NLP heuristics that we implemented in order for the tool to determine which speaker turns are relevant to a given requirement.
- Through collaboration with a partner in the software consultancy, we present an early validation of our implemented algorithms on automatically generated transcripts of requirements interviews.

We provide qualitative observations on search strategies in backward traceability, we reflect on goals & use-cases, and we position initial approaches for ranking relevance. Additionally, the reported validation serves not only as a formative evaluation that guides the evolution of TRACE2CONV, but also as a kick-off for future research in the field of *conversational RE*.

Organization. We motivate our work with reference to the existing literature in Sec. II. Then, we present the design of TRACE2CONV and provide an overview in Sec. III. We report on our validation in Sec. IV. Finally, we present a discussion and outline future directions in Sec. V.

II. Motivation & Related Work

Pre-RS traceability aims to establish trace links between the needs in the problem space and the requirements in the solution space [12]. Ravichandar *et al.* [12] position the Capabilities Engineering approach as a glue between problem and solution space. These capabilities can be used to define the requirements as a high-level architecture while remaining abstract enough to handle detailed changes in the requirements.

In the work on abstraction identification by Gacitua *et al.* [13], the authors propose an approach to provide requirements engineers with abstractions (domain terms) to support knowledge gathering. Although not directly executing pre-RS tracing, the use of external documentation to identify domain terms is a relevant activity for understanding which domain elements the requirements may operate on, and these terms can also provide additional information for the interpretation of a requirements conversation.

Requirements conversations have not been a common object of RE research. Alvarez and Urla [6] conducted one of the very few studies that investigate the narrative structure behind requirements conversations. They study requirements interviews conducted as part of the implementation of a large-scale ERP system, and they identify the important role of the stories that the clients and stakeholders tell. Their interesting findings, however, are based on a time-consuming manual analysis of the interviews transcripts, which is feasible for research but inadequate for practical RE settings.

Ferrari *et al.* [14] focus on RE interviews in their work on voice and biofeedback to identify the emotions of the speaker. They find that voice analysis alone can lead to good-enough results, without the necessity of using the more intrusive biofeedback analysis. Similar findings were presented in the Google Cloud Next 2019 [15] conference, where sentiment analysis was used in a call-center scenario to review customer satisfaction. In this version of our work, we focus on text transcripts generated from interview recordings. Sentiment and emotion analysis are intriguing future directions.

In our research, we make use of speaker turns, which is a form of speaker diarization that is part of conversation analysis [16]. The automated transcription services we use offer this feature to provide the transcript in a format that is split into these speaker turns. The services can recognize automatically when the speaker turn changes by detecting differences in voice, different audio streams, and speech events like silence to identify utterances. Park *et al.* [17] cover the history and advances of speaker diarization.

Our work focuses on the use of transcripts from requirements elicitation sessions. While we see these as a valuable asset and initial source of information, it is also important to acknowledge the limitations of the artifacts. Ferrari *et al.* [8] note a number of ambiguity types and their effects in interviews. These can all negatively impact the value of the information that we present to the user. Especially users who were not present in the conversations might take a segment of it as truth, disregarding the limitations of the conversational setting. Additionally, requirements evolve between the discussions in the recordings and the authoring of the requirements specification. In recent research, Debnath *et al.* [18] found that 30-38% of finalised requirements can be fully traced to the initial specification, 54-63% are refinements, and 13-21% are completely novel. We expect a similar evolutionary refinement of the discussion into the creation of the specification.

III. Trace2Conv: Design and Overview

For the design of Trace2Conv, we started with a manual review of two data sets of real-world RE interviews for two information systems. One of these data sets concerned a system supporting core operating processes, and the recording consisted of an 1.5 hours on-site session and a 2 hours digital conversation with 4-5 participants. The second data set regarded a web portal to externalize data from an ERP, and was discussed in a 1.5 hour digital session with 8 participants. In this analysis, we manually tagged automatically transcribed interviews in NVivo to identify requirements-relevant information and to define the goals to be achieved by Trace2Conv. During this tagging, the transcripts had an average of 44.5% of the transcript text tagged as requirements relevant, which indicates not only a summarization opportunity, but also potential for tool-assisted exploration of the transcript to enable effectively finding this relevant information.

The main use cases that we investigate in this research relate to *increasing the available context for existing requirements and linking them to parts of the requirements conversation*. Given a set of requirements R – consisting of user stories in this paper – and a requirements conversation C – an interleaving of speaker turns uttered by multiple speakers – we identify the following scenarios: (i) developers can review parts of a conversation they otherwise wouldn't for additional information on their assigned tasks; (ii) requirement engineers can use the related speaker turns for their initial requirements set to enhance their requirements specifications; and (iii) project managers can trace back requirements to the conversation to identify agreements made with the client.

Although these use cases can be seen as a traceability problem, pre-RS traceability poses additional challenges that make existing techniques from the literature unsuitable:

1) While each requirement is an individual element to be traced, there is no distinguishable, atomic element in a conversation. We aim to provide *additional context* to a requirement, rather than only finding a specific speaker turn or utterance where that requirement is stated (this distinction is elaborated upon later, in Table I).
2) Requirements conversations can be informal, unstructured [19], and they include jargon that does not follow the quality conventions that are often used for writing requirements [20], for naming classes and methods, etc.
3) The process of writing a specification from requirements interviews includes an *abstraction gap*–also called complexity gap [21], [12]–from the problems that are discussed in the interviews to the functions and constraints that are set for the system to-be.

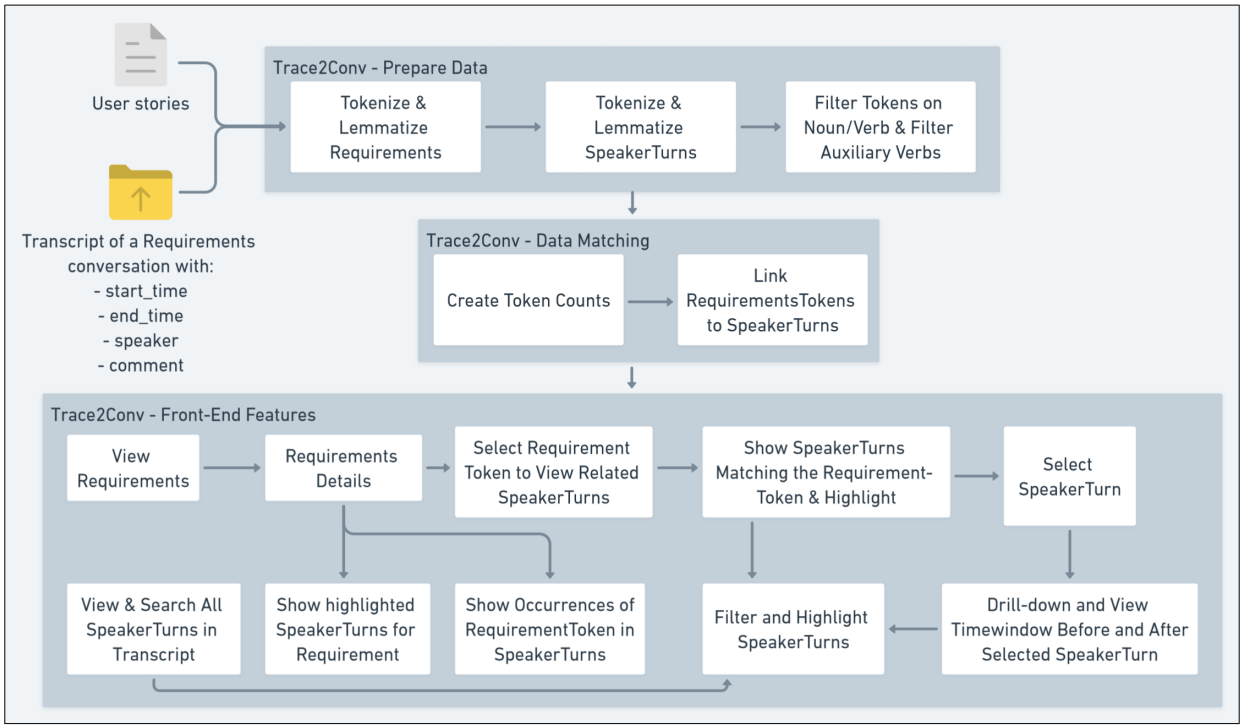

Fig. 1. Matching process & front-end features of TRACE2CONV

Therefore, our solution, sketched in Fig. 1, focuses on identifying potential matches between requirements and speaker turns in a conversation, to facilitate the exploration of a transcript and gain further context for a requirement, rather than establishing the actual trace links between the artifacts.

Pre-processing. The role of pre-processing activities, highlighted in Fig. 2, is to convert an audio recorded requirements elicitation conversation into a textual representation.

Since our use cases assume the existence of a set of user story requirements R, we build on previous research to improve the transcript recognition for the automated speech recognition software (for this study, AWS Transcribe [22]). Preprocessing starts by analyzing a collection of user stories using the Visual Narrator tool [23], which outputs a domain model with the most important concepts in the user stories. These concepts are used as a custom vocabulary for AWS Transcribe, to improve the accuracy of transcript and, thus, to be better able to match the requirements to the transcript segments. The requirements conversation recording is then converted to a format supported by AWS, and uploaded. This is run as a transcription job in AWS transcribe and the output is downloaded and then transformed into an CSV file with Python's `tscribe` [24]. Finally post-processing is conducted to combine adjacent speaker turns uttered by the same speaker.

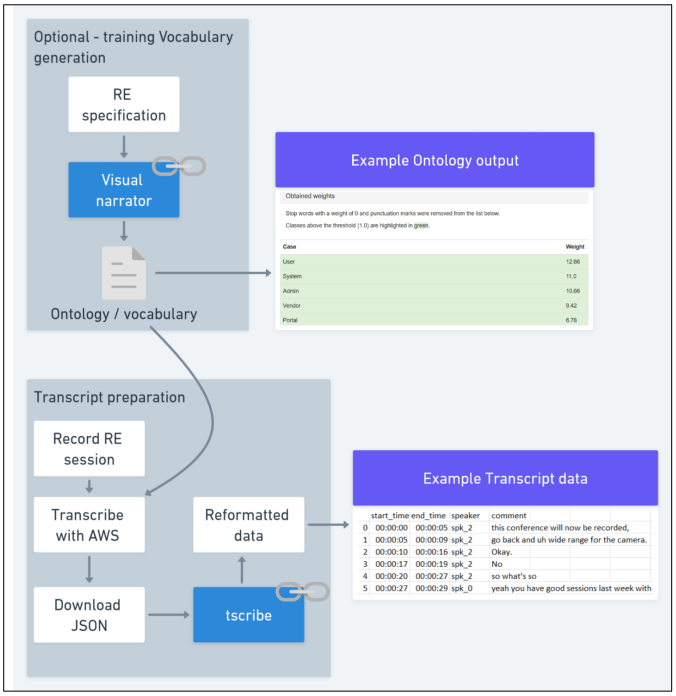

Fig. 2. Pre-processing process for TRACE2CONV leading to the transcript file

Processing After the pre-processing steps are completed, TRACE2CONV can initiate the process of matching the user stories with the transcript of a requirements conversation. In our current prototype, the requirements are inputted from a CSV file, each line containing one user story. The transcript is a CSV file where every line is a speaker turn with information about the start time, end time, speaker, and uttered text.

First, TRACE2CONV executes the steps detailed in Algorithm 1: both transcript and requirements are tokenized and lemmatized (lines 3–4), then the identified tokens are filtered

depending on the POS tag. Based on our empirical analysis, this first version of TRACE2CONV includes tokens that are either nouns or verbs (the list of tokens T_{sent} is reduced in lines 5–6), excluding auxiliary verbs. Then, for each identified token (line 8), TRACE2CONV creates the set of requirements and speaker turns where that token occurs (lines 9–10).

Algorithm 1 Data Preparation and Data Matching

Input: R a set of requirements,
C a set of speaker turns,
Output: a set of tokens $AllTokens$, linked to their occurrences in requirements and speaker turns

1: **function** PREPROCESSANDMATCH(R, C)
2: **for all** $sent \in R \cup C$ **do**
3: $T_{sent} \leftarrow$ TOKENIZE($sent$)
4: $T_{sent} \leftarrow$ LEMMATIZE(T_{sent})
5: $T_{sent} \leftarrow (t \in T_{sent} \mid \text{POS\_TAG}(t) \in \{\texttt{NOUN}, \texttt{VERB}\})$
6: $T_{sent} \leftarrow (t \in T_{sent} \mid \text{POS\_TAG}(t) \notin \{\texttt{AUX}\})$
7: $AllTokens \leftarrow \bigcup_{sent \in R \cup C} T_{sent}$
8: **for all** $t \in AllTokens$ **do**
9: $t.reqs \leftarrow \{req \in R \mid t \in T_{req}\}$
10: $t.turns \leftarrow \{sturn \in C \mid t \in T_{sturn}\}$
11: **return** $AllTokens$

Algorithm 2 takes care of determining the counts that show the frequency of the tokens. After calling Algorithm 1 in line 2, three counts are calculated: the number of requirements where that token occurs (line 4), the number of turns where the token occurs (line 5), and the number of total occurrences of that token in the requirements conversation (line 6).

Algorithm 2 Token Counts

Input: R a set of requirements,
C a set of speaker turns

1: **function** TOKENCOUNT(R, C)
2: $T \leftarrow$ PREPROCESSANDMATCH(R, C)
3: **for all** $tk \in T$ **do**
4: $tk.reqCount \leftarrow |\{r \in R \mid tk \in T_r\}|$
5: $tk.turnCount \leftarrow |\{sturn \in C \mid tk \in T_{sturn}\}|$
6: $tk.instCCount \leftarrow |\bigcup_{sturn \in C}\{t' \in T_{sturn} \mid t' = tk\}|$
7: **return** T

User Interface. The outputs of the processing are used by the user interface, where the user can select their transcript to review the speaker turns in order of occurrence with additional manual highlighting and filtering. The matching process that links requirements tokens and conversations tokens enables the most valuable functionality of TRACE2CONV: the ability to drill down from a requirement into the speaker turns where certain key terms are discussed, providing extra context and tracing the requirement to the RE interview. In the current prototype, this exploration process is led by the user, who decides which terms to focus on; TRACE2CONV does provide counts on the frequency of the term in the requirements and in the speaker turns, as well as the number of speaker turns where the term occurs: see Fig. 3.

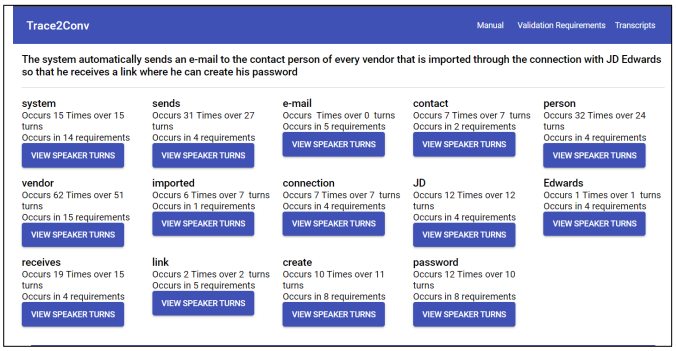

Fig. 3. Requirements Token Review in TRACE2CONV

The user can then select one of the requirements tokens to drill-down into the speaker turns that match the requirement token as seen in Fig. 4. For guiding the user, the requirement token are highlighted in the speaker turns. By clicking view on one specific speaker turn, the user can review the speaker turn in the wider conversation context, and is presented with the 5 minutes of conversation before and after the speaker turn.

In the roadmap (Section V), we discuss improved algorithms based on our validation to suggest speaker turns for the requirements to further support locating relevant information.

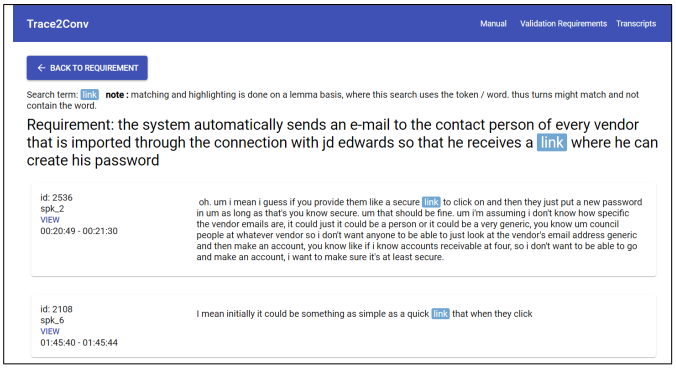

Fig. 4. Matching Speaker Turns for a Requirement Token in TRACE2CONV

IV. VALIDATION

We performed a formative [25], post-implementation [26] evaluation of our prototype designed based on the guidelines from the *Framework for Evaluation in Design Science* (FEDS) by Venable *et al.* [27]. This framework splits the evaluation approach into four steps, which we apply to our research.

Step 1: Explicate the goals. In this evaluation, we validate the initial results provided by the tooling, and we observe how an RE expert utilizes current functionality, their search strategies and whether consistent behavior can be observed. Our aim is to inform the design of algorithms that mirror the observations.

Step 2: Choose a strategy for the evaluation. We utilize a "quick and easy" [27] strategy for this first validation. We

aim to discover ways to improve the tooling and to evaluate initial results. We opt for an evaluation strategy that is executed without relying on external validation. Three reviewers took part in the evaluation: two are authors of this paper, and the third reviewer is conducting research on a similar topic.

Step 3: Determine the properties to evaluate. We focus on four properties: (i) search strategies based on the presented tokens; (ii) initial feedback on using the tool; (iii) review of requirement token evaluation types; (iv) consistency of the different reviewers for the selection of search terms.

Step 4: Design the individual evaluation episodes. First, shared understanding of a specific case was ensured. The system discussed in the transcript was demoed and the domain terms were discussed and highlighted. Given a requirement r and a speaker turn st, we set the guidelines and examples for speaker turn relevance that are shown in Table I.

TABLE I
GUIDELINES FOR DETERMINING THE RELEVANCE OF A SPEAKER TURN *st* FOR A REQUIREMENT *r*.

Guideline 1	*st* elaborates on a domain concept that appears in r.
Example r	The system will initiate transfer from a bulk location to a picking location if an order exceeds the current picking location stock.
Spkr turn st	"So we utilize 2 sort of stocks, those directly accessible to our order pickers to gather in their lists, and harder to reach stock. These are kept on the highest shelves in a bulk location, and require a forklift to move."
Guideline 2	*st* explicitly discusses the functionality in r.
Example r	As a user, I can assign a task with a deadline to someone else, so that we can ensure follow up and appropriate notifications.
Spkr turn st	"We're working with action items, these are just a message to notify a specific user, what we'd like is a more actionable way of keeping track of these. Using a date to complete the action item, we can also provide our managers an overview with tasks they allocated, and give them a heads-up when something is overdue."
Guideline 3	*st* provides rationale or additional background for r, although it does not explicitly request r.
Example r	As an admin, I want to approve new supplier requests, so that no data is entered into our ERP without a validation.
Spkr turn st	"We get a huge amount of supplier requests, so in a week we might be adding more than 20 new suppliers. We have a unit of people working on this in our company. So they just receive supplier application forms, and key in all the data in the correct places."

The reviewers first individually tagged the same three requirements, determining the tokens to review, and tagging speaker turns on relevance. Afterwards, the results and reasoning for tagging were discussed, leading to the final guidelines of Table I. Then, each reviewer individually tagged the remaining 27 requirements and took notes on the review, as a qualitative evaluation.

Evaluation results. Out of the total 369 requirement tokens returned by TRACE2CONV, the reviewers evaluated 112 (30.35%), 107 (29.00%) and 73 (21.41%) of the tokens, respectively. The set of tokens contained 250 nouns (67.7%) and 119 verbs (32.3%); each of the reviewers had a similar distribution: 66.7%, 71.9%, 68.6% of the tags were nouns. We did not detect any major difference in the POS tags for the tokens compared to the general distribution. For the 149 unique tokens tagged, 49 were shared among all reviewers, 44 were reviewed by 2 reviewers and 56 were reviewed by a single reviewer: 50.52% ($(49 \times 3/(112 + 107 + 73)$), 30.24% and 19.24% of all tags, respectively. This indicates a similar search strategy by the three reviewers.

Qualitative observations. The free-form notes taken by the reviewers include observations on the use of TRACE2CONV and possible improvements to the user experience.
While the reviewers tested multiple *search strategies* throughout the process, the main strategy was the following: (i) tag the action verbs and the main nouns in the requirement while trying to prevent selecting those occurring too often; (ii) if no results were found in the initial selection, the terms with higher occurrence would be considered.

Similar and overlapping requirements were also influencing factors: when exploring a similar requirement, the reviewer already knew where to look and could quickly find the matching speaker turns. Additionally, while the prototype did not support it, all reviewers mentioned using their browser's *search functionality* to further filter the resulting speaker turns and to find compound nouns (e.g., "vendor user"). The reviewers observed that the context influences the search process. For instance, for a requirement regarding an API connection, it was noted that the information around the invoices was more interesting than the term "connection" itself.

In addition to the increasing familiarity of the resulting turns, which influenced this process, the reviewers also reported *process fatigue*. Over time, the aim of identifying all relevant speaker turns evolved into finding a set of matching segments that would satisfy the reviewer. This sets high expectations on the precision and recall for the automation. However, it is worth noting that our validation use case differs from a real use case, in which we expect the analyst to investigate a few requirements instead of a full set.

As hypothesized, *not all requirements can be traced*. Multiple reasons were mentioned: some requirements were detailing standard technical functionality (e.g., password length constraints), some were clearly written by the analysts for specifying tasks for the developers to find in their backlog. Similarly, some requirements are detailed enough not to require any tracing back to their source; for instance, a requirement that describes the need to show the username. The tool could be improved by showing not only possible matching speaker turns, but also the *confidence*, or likelihood, of relevance.

Some challenges depend on the quality of the artifacts. Requirements sometimes use specific *implementation terms*, e.g., "export as xls file", while conversations would use the generic application name. The tooling could use a *glossary* of common terms to expand the search in these cases and thus also search for mail, email and message when a requirement contains "e-mail". Importantly, the results were impacted by the quality of the *automated transcriptions*. While a human with context knowledge might interpret "fender" as "vendor"

(these words a similar prononuciation, as they share the metaphone [28]: FNTR), our search and matching processes do not. We might have to support ways to fix transcripts, or expect the user to improve them before importing.

V. Roadmap and conclusions

After the validation, we devised some improvements of TRACE2CONV aimed to highlight speaker turns that are likely relevant, with the aim of providing a prioritized list of speaker turns for the analyst to examine, and avoid process fatigue that may arise by exploring each of the tokens. We devised and implemented two prototype ranking mechanisms: *single-token occurrence* & *multiple-token occurrence*. In the current design iteration, these mechanisms are preliminary and are to be improved through validation and design iterations.

Single-token occurrence lists speaker turns that contain one of the requirement tokens 2+ times. For instance, a speaker turn that contains the lemma for "vendor" 3 times could be *"Our vendors will be logging in to the application, and then they would be able to see the vendor information, and request updates to their information. This would then end up in our mailbox as a vendor request."*. Future refinements of this mechanism could assign different weights based on the sentence structure, and token tags, minimizing false positives.

Multiple-token occurrence lists speaker turns based on a ranking mechanism that counts how many of the collection of requirement token lemmas occur in a particular speaker turn. For example, given the tokens [Doctor, prescribe, medicine, headache, dashboard, paracetamol] the following speaker turn would have a score of 7: *"Our doctors[1] prescribe[2] all kinds of medicine[3], however, it can be hard to keep track of all the things that have been historically prescribed[4] to a patient. Thus what we're looking for is some sort of dashboard[5] that presents at a glance what has been prescribed[6] to a patient by any of our doctors[7]."* . Also for this mechanism, we are looking into weighted ranking versions.

Additional relevance variables. We consider the following items as potential variables that may improve the precision of the results. The *location* of the terms in the requirement can indicate importance: lower priority can be given to the roles and the so-that part of a user story. The *occurrence count* of the terms in the speaker turns should be considered, since we observed that a lower total occurrence increases the chances of a match. *Speaker turn length* will impact the ranking mechanisms: longer speaker turns are also more likely to contain multiple terms, and one may need to normalize the score based on the length of the speaker turns. The *surrounding speaker turns* may also be important to determine the likeliness of relevance of a single turn.

Other tool enhancements. While we utilize low-code for early prototyping, we acknowledge its limitations. Many *NLP libraries* that can easily be used in Python are unavailable in our current prototype. We are currently working toward using standard NLP techniques for trace link recovery including language patterns, information retrieval (e.g., TF-IDF) and machine learning classifiers. To do this, we plan to devise a collection of API endpoints that can be called by TRACE2CONV. A more ambitious direction that we are pursuing is the identification of relevant speaker turns without having an existing set of requirements, extending the use case to supporting the creation of the requirements specification.

Additionally, the tool will be extended to support using it over *multiple transcripts*. Since most requirements elicitation in projects is done over multiple conversations, and requirements change over time [18], TRACE2CONV will support a project database of both transcripts and requirements that can evolve over time. This also introduces recency as a variable in speaker turn suggestions, potentially allowing the user to define their use case to effectively present results.

A next iteration of the tooling will also include the review of *heat zones*, segments of a transcript with multiple matches (e.g., between minute X and Y) that are more likely to be of interest to the user, including the unmatched turns.

Overall, while we discuss different approaches to do so, we aim to suggest and trace mainly on a requirement level instead of on a requirement token level, as it removes steps the user would need to take and creates consistency in the results. The current token-based functionality can still be offered as drill-down functionality when suggestions are inconclusive.

One major limitation is transcript quality, whose word-error-rates are getting smaller. While we could leave it up to the users to provide correct transcripts, we will look into ways to improve the automated transcript accuracy, including approaches for the interactive correction.

Validation. For the next phases of our research, we have planned a more extensive validation with external stakeholders. However, our preference is to first further refine the tool to get a more significant validation. We currently envision the following validation protocol; a set of potential users will be asked to use the application based on a domain they are familiar with. They will then be asked to perform a set of tasks, e.g., finding the context or source of a particular requirement, then reflect on and evaluate the use of the tool. Also, we plan to perform more validations on the resulting traceability and compare our token based approach to common NLP techniques like TF-IDF [29] and vector space models [30].

Conclusion. This paper introduced the notion of conversational RE. Furthermore, we proposed a first attempt to establish backward traceability from requirements to one or more relevant transcript segments in a requirements conversation with TRACE2CONV. This is done by matching speaker turns to the requirements on a tokenization and lemmatization basis. While functionality and UI need further development for use in practice, our preliminary results show the feasibility of the overall approach, which makes use of low-effort technologies to achieving backward traceability.

Through further design iterations, we aim to support practitioners by providing them with a backward traceability of

requirement documents to conversations without enforcing a change in work methods. This can improve the information available to analysts during their projects, but also when we consider employee turnover and application maintenance. Another benefit is that TRACE2CONV can be used as a repository for requirements and transcripts within an organization.

ACKNOWLEDGMENT

The authors would like to thank Xavier de Bondt for his help on tagging the data, Kosmas Ntokos for his help with developing TRACE2CONV, and the members of the RE-Lab at Utrecht University for the valuable discussions on the research.

REFERENCES

[1] O. C. Gotel and C. Finkelstein, "An analysis of the requirements traceability problem," in *IEEE International Conference on Requirements Rngineering*, 1994, pp. 94–101.

[2] J. Krause, A. Kaufmann, and D. Riehle, "The code system of a systematic literature review on pre-requirements specification traceability," https://doi.org/10.25593/issn.2191-5008/CS-2020-02, Tech. Rep., 2020.

[3] A. De Lucia, F. Fasano, and R. Oliveto, "Traceability management for impact analysis," in *Frontiers of Software Maintenance*. IEEE, 2008, pp. 21–30.

[4] P. Heck and A. Zaidman, "Horizontal traceability for just-in-time requirements: the case for open source feature requests," *Journal of Software: Evolution and Process*, vol. 26, no. 12, pp. 1280–1296, 2014.

[5] A. Ferrari, G. O. Spagnolo, and S. Gnesi, "PURE: A dataset of public requirements documents," in *IEEE International Requirements Engineering Conference*. IEEE, 2017, pp. 502–505.

[6] R. Alvarez and J. Urla, "Tell me a good story: Using narrative analysis to examine information requirements interviews during an ERP implementation," *ACM SIGMIS Database: the DATABASE for Advances in Information Systems*, vol. 33, no. 1, pp. 38–52, 2002.

[7] T. Spijkman, F. Dalpiaz, and S. Brinkkemper, "Requirements elicitation via fit-gap analysis: A view through the grounded theory lens," in *International Conference on Advanced Information Systems Engineering*. Springer, 2021, pp. 363–380.

[8] A. Ferrari, P. Spoletini, and S. Gnesi, "Ambiguity and tacit knowledge in requirements elicitation interviews," *Requirements Engineering*, vol. 21, no. 3, 2016.

[9] M. Bano, D. Zowghi, A. Ferrari, P. Spoletini, and B. Donati, "Teaching requirements elicitation interviews: an empirical study of learning from mistakes," *Requirements Engineering*, vol. 24, no. 3, pp. 259–289, 2019.

[10] M. Cohn, *User stories applied: For agile software development*. Addison-Wesley Professional, 2004.

[11] G. Lucassen, F. Dalpiaz, J. M. E. van der Werf, and S. Brinkkemper, "The use and effectiveness of user stories in practice," in *International Working Conference on Requirements Engineering: Foundation for Software Quality*. Springer, 2016, pp. 205–222.

[12] R. Ravichandar, J. D. Arthur, and M. Pérez-Quinones, "Pre-requirement specification traceability: Bridging the complexity gap through capabilities," *arXiv preprint cs/0703012*, 2007.

[13] R. Gacitua, P. Sawyer, and V. Gervasi, "On the effectiveness of abstraction identification in requirements engineering," in *18th IEEE International Requirements Engineering Conference*. IEEE, 2010, pp. 5–14.

[14] A. Ferrari, T. Huichapa, P. Spoletini, N. Novielli, D. Fucci, and D. Girardi, "Using voice and biofeedback to predict user engagement during requirements interviews," *arXiv preprint arXiv:2104.02410*, 2021.

[15] Google, "Optimizing voice commands and IVRs to speech analytics (Cloud Next '19)," https://www.youtube.com/watch?v=71jzm19xn4U.

[16] P. Drew, "Conversation analysis," in *Handbook of Language and Social Interaction*, P. Drew and J. Heritage, Eds., 2005, vol. 2, no. 3, pp. 71–102.

[17] T. J. Park, N. Kanda, D. Dimitriadis, K. J. Han, S. Watanabe, and S. Narayanan, "A review of speaker diarization: Recent advances with deep learning," *Computer Speech & Language*, vol. 72, p. 101317, 2022.

[18] S. Debnath, P. Spoletini, and A. Ferrari, "From ideas to expressed needs: an empirical study on the evolution of requirements during elicitation," in *IEEE International Requirements Engineering Conference*. IEEE, 2021, pp. 233–244.

[19] R. Torkar, T. Gorschek, R. Feldt, M. Svahnberg, U. A. Raja, and K. Kamran, "Requirements traceability: a systematic review and industry case study," *International Journal of Software Engineering and Knowledge Engineering*, vol. 22, no. 03, pp. 385–433, 2012.

[20] IEEE, "IEEE recommended practice for software requirements specifications," IEEE Std 830-1998, Tech. Rep., 1998.

[21] L. Raccoon, "The complexity gap," *ACM SIGSOFT Software Engineering Notes*, vol. 20, no. 3, pp. 37–44, 1995.

[22] Amazon, "Amazon transcribe: custom-vocabularies." [Online]. Available: https://docs.aws.amazon.com/transcribe/latest/dg/custom-vocabulary.html

[23] G. Lucassen, M. Robeer, F. Dalpiaz, J. M. Van Der Werf, and S. Brinkkemper, "Extracting conceptual models from user stories with Visual Narrator," *Requirements Engineering*, vol. 22, no. 3, pp. 339–358, 2017.

[24] Github, "tscribe, AWS transcribe to docx," 2022. [Online]. Available: https://github.com/kibaffo33/aws_transcribe_to_docx

[25] D. Remenyi and M. Sherwood-Smith, *IT Investment: Making a business case*. Routledge, 2012.

[26] S. Smithson and R. Hirschheim, "Analysing information systems evaluation: another look at an old problem," *European Journal of Information Systems*, vol. 7, no. 3, 1998.

[27] J. Venable, J. Pries-Heje, and R. Baskerville, "FEDS: a framework for evaluation in design science research," *European Journal of Information Systems*, vol. 25, no. 1, 2016.

[28] L. Philips, "Hanging on the metaphone," *Computer Language*, vol. 7, no. 12, pp. 39–43, 1990.

[29] R. Baeza-Yates, B. Ribeiro-Neto *et al.*, *Modern information retrieval*. ACM press New York, 1999, vol. 463.

[30] G. Salton, A. Wong, and C.-S. Yang, "A vector space model for automatic indexing," *Communications of the ACM*, vol. 18, no. 11, pp. 613–620, 1975.