Pinpointing Ambiguity and Incompleteness in Requirements Engineering via Information Visualization and NLP

Fabiano Dalpiaz[0000−0003−4480−3887], Ivor van der Schalk, Garm Lucassen[0000−0002−0213−0699]

RE-Lab, Dept. of Information and Computing Sciences, Utrecht University
{f.dalpiaz,i.l.vanderschalk,g.lucassen}@uu.nl

Abstract. **[Context and motivation]** Identifying requirements defects such as ambiguity and incompleteness is an important and challenging task in requirements engineering (RE). **[Question/Problem]** We investigate whether combining humans' cognitive and analytical capabilities with automated reasoning is a viable method to support the identification of requirements quality defects. **[Principal ideas/results]** We propose a tool-supported approach for pinpointing terminological ambiguities between viewpoints as well as missing requirements. To do so, we blend natural language processing (conceptual model extraction and semantic similarity) with information visualization techniques that help interpret the type of defect. **[Contribution]** Our approach is a step forward toward the identification of ambiguity and incompleteness in a set of requirements, still an open issue in RE. A quasi-experiment with students, aimed to assess whether our tool delivers higher accuracy than manual inspection, suggests a significantly higher recall but does not reveal significant differences in precision.

Keywords: Natural Language Processing, Requirements Engineering, Information Visualization, User Stories, Ambiguity.

1 Introduction

Defects in natural language (NL) such as ambiguity, unclarity, inconsistency, and incompleteness are common issues in requirements engineering (RE) [1,2,3], and they can lead to misunderstandings between stakeholders, overlooked requirements, and software systems that do not meet the stakeholders' needs.

The identification of requirements defects is no trivial task. Automated solutions are inhibited by the low maturity of NL processing (NLP) techniques—unable to gain a deep understanding of text [4]—and the necessary trade-offs between precision and recall [5,2,6]. On the other hand, manual approaches that rely on human intelligence and the application of inspection checklists, do not scale to large specification. Luckily, the two approaches are not incompatible.

We make a step toward the synergistic use of NLP and human analysis as part of our research on user stories and agile RE. User stories are semi-structured

notation for user requirements with a simple format [7]: *As a* student, *I want to* receive my grades via e-mail, *so that* I can quickly check them. We take as input the terms and relationships that are automatically extracted by our Visual Narrator tool [8] from a set of user stories. Unfortunately, despite its high extraction accuracy, Visual Narrator does not assist analysts to inspect the resulting graphical model, thereby making our approach impractical for large models.

In this paper, we modularize the models extracted from user story requirements by leveraging the viewpoints [9] that user stories natively express through their format (As a user ... ; As a developer ...). Such approach is embedded in a Web 2.0 tool that blends NLP and information visualization (InfoVis) techniques with the aim of identifying potential ambiguities and missing requirements.

We make four concrete contributions:

- We construct a framework that defines potential ambiguity and incompleteness based on the terminology and denotations used in different viewpoints.
- We build an algorithm for identifying (near-)synonyms that orchestrates state-of-the-art semantic similarity algorithms from the NLP domain.
- To help analysts explore potential defects, we propose a Venn diagram visualization that organizes the extracted terms according to the viewpoint(s), and emphasizes terminological ambiguity using colors.
- We report on a quasi-experiment that assesses whether pairs of analysts using the tool on a large interactive screen obtain higher precision and recall in identifying quality defects than analysts working pen-on-paper.

Organization. We explain our framework for identifying ambiguity and incompleteness starting from viewpoints in Sec. 2, then present the algorithm for detecting (near)-synonymy ambiguity in Sec. 3. We introduce our Venn diagram visualization in Sec. 4. We report on the evaluation in Sec. 5, discuss related work in Sec. 6, draw conclusions and present future directions in Sec. 7.

2 From Viewpoints To Ambiguity and Incompleteness

The different stakeholders of a software system are interested in distinct aspects. For example, website administrators care about content creation and structuring, while readers are mostly concerned in accessing existing content. According to Mullery [10], a *viewpoint* is a description of one stakeholder's perception of a system, and it consists of concepts and inter-relationships between them.

The existence of viewpoints inevitably leads to inconsistencies and conflicts in stakeholders' requirements. Recognizing and reconciling these issues are key tasks in RE [11], and they amount to (i) checking the consistency of the specification within one viewpoint (in-viewpoint checks), and (ii) checking the consistency of the specification among different viewpoints (inter-viewpoint checks) [9].

Viewpoints may also introduce ambiguity problems due to the use of different terminology and conceptual systems (how an expert assigns meaning to a term [12]). The descriptions of a domain by different experts lead to four types of relationships that depend on their chosen terminology (bank, car) and the

distinctions (also known as *denotations*) in the domain that the terms refer to (a financial institution, a ground alongside a body of water, a road vehicle) [12]:

1. *Consensus*: same terminology, same distinction. Example: both experts use the term bank to refer to a financial institution.
2. *Correspondence*: different terminology, same distinction. Example: when referring to a road vehicle, one expert uses car and the other uses automobile.
3. *Conflict*: same terminology, different distinction. Example: both experts use bank, but one refers to a financial institution, while the other to a ground.
4. *Contrast*: different terminology, different distinction. Example: one viewpoint examines road vehicles, the other focuses on financial institutions.

A requirement is ambiguous when it has multiple valid interpretations [13]. We argue that when a collection of requirements contains terms related by correspondence or conflict, there is a possible ambiguity. Furthermore, possible missing requirements may arise due to contrast. Table 1 formalizes these concepts.

Table 1: Linking viewpoints' terminological and denotational relations [12] with possible ambiguity and incompleteness. Let t_1, t_2 be distinct terms, $[\![t]\!]^{V_1}$ be the denotation of term t according to the viewpoint V_1 (for simplicity, we assume that denotations refer to a single entity), and $\perp$ indicate absence of a denotation.

Relation [12]	Possible defect	Defect formalization	Example
Consensus	-	$[\![t_1]\!]^{V_1} = [\![t_1]\!]^{V_2}$	$[\![\text{bank}]\!]^{V_1}$ = financial institution $[\![\text{bank}]\!]^{V_2}$ = financial institution
Correspondence	(Near-)synonymy leading to ambiguity	$[\![t_1]\!]^{V_1} = [\![t_2]\!]^{V_2}$	$[\![\text{car}]\!]^{V_1}$ = road vehicle $[\![\text{automobile}]\!]^{V_2}$ = road vehicle
Conflict	Homonymy leading to ambiguity	$[\![t_1]\!]^{V_1} \neq [\![t_1]\!]^{V_2}$	$[\![\text{bank}]\!]^{V_1}$ = financial institution $[\![\text{bank}]\!]^{V_2}$ = land alongside river
Contrast	Incompleteness	$[\![t_1]\!]^{V_1} \neq \perp \wedge [\![t_1]\!]^{V_2} = \perp$	$[\![\text{bank}]\!]^{V_1}$ = financial institution $[\![\text{bank}]\!]^{V_2} = \perp$

Consider now an example: take the following four user stories from the WebCompany data set [8] (terms are emphasized in serif):

R_1. As a visitor, I am able to view the media gallery, so that I can see interesting photos about the event region.
R_2. As an administrator, I am able to edit existing media elements of a particular gallery, so that I can update the content.
R_3. As a user, I am able to add content to the selected profile.
R_4. As a visitor, I am able to use the contact form, so that I can contact the administrator.

Consensus does not lead to any ambiguity. For example, the term administrator has the same denotation both in R_2 and R_4 and it refers to the person managing the website and its users.

Ambiguity *may* occur with correspondence: distinct terms refer to the same denotation. The term media gallery in R_1 and the term gallery in R_2 do likely (but

not necessarily) refer to the same denotation, a web gallery where photographs are displayed. The problem is that most synonyms are in fact near-synonyms (*plesionyms*), as they refer to similar yet not identical denotations [14].

Ambiguity *may* also occur in the conflict state: the same term is used for different denotations. This phenomenon is called *homonymy*. In R_2, the term content refers specifically to a media element, while in R_3 the term content may refer to either text, descriptions, images, videos or audio fragments.

Incompleteness (missing requirements) *may* occur in the contrast state, i.e., in the case in which one viewpoint refers to concepts that do not appear in another viewpoint. R_4 includes contact form that the visitor uses to get in touch with the administrator. However, there is no other user story in our short collection that specifies how the administrator can respond to this action.

3 NLP-Powered Identification of (Near)-Synonymy

To detect *(near)-synonymy* between terms that may lead to ambiguity (the *correspondence* relationship in Table 1), we develop an NLP-powered algorithm that integrates state-of-the-art semantic similarity techniques. This algorithm is used in Sec. 4 to set the terms' background color in the InfoVis approach.

Our NLP technique relies on algorithms that calculate the *semantic distance* between two terms: a numerical representation of the difference in meaning between two terms [15]. Current state-of-the-art NLP tools, such as Word2Vec, establish semantic similarity in the $[0.0, 1.0]$ range via word statistics that compare the contexts in which a term is used [16]. The higher the similarity score, the higher the chance that the two terms have the same denotation.

In this paper, we invoke the Cortical.io[1] tool that employs Semantic Folding Theory (SFT), a novel method that creates sparse distributed representations of terms (their semantic fingerprint [17]). Each activated bit of the semantic fingerprint represents a characteristic of that word. For example, some of the activated bits for the word *dog* may denote the concepts *fur*, *barking*, *omnivore*, while some activated bits for the word *moose* may represent *fur*, *herbivore*, *horn*. The higher the number of shared activated bits, the higher the similarity between two words.

Algorithm 1 takes a set of user story requirements and generates an ambiguity score for all couples of terms that appear in the use stories. In line 1, the Visual Narrator tool [8] extracts nouns (e.g., car, dog) and compound nouns (e.g., cable car, sledge dog) from the set *userStories*. Then (line 2), all combinations of term pairs are added to the variable *termPairs*. The algorithm constructs the context of each term (lines 3–5), i.e., the set of all user stories that contain such term.

The loop of lines 6–12 takes care of computing the ambiguity score for each pair of terms $(t1, t2)$. The semantic similarity of the two terms is computed in line 7; we use the Cortical.io algorithm based on semantic folding and fingerprints. Then, the algorithm builds the context of each term pair: all and only the

[1] http://api.cortical.io/

user stories where exactly one of the two terms occurs (lines 8–10). We exclude the user stories where both terms occur because we assume that the analyst who writes a story purposefully chooses the employed terms, and therefore two distinct terms in the same story are unlikely to be in a correspondence relation.

The similarity score can now be determined–again, via Cortical.io–for the contexts of each pair of terms (line 11). Finally, the ambiguity score (line 12) is computed as a linear combination of term similarity and context similarity. We currently assign a weight of 2 to former and a weight of 1 to the latter.

Algorithm 1 Computing the (near)-synonymy ambiguity score of term pairs

ComputeAmbigScore(Set⟨UserStory⟩ *userStories*)

1 Set⟨Term⟩ $usTerms$ = VisualNarrator($userStories$)
2 (Term,Term) $termPairs = (t1, t2).\ t1, t2 \in usTerms \wedge t1 \neq t2$
3 Set⟨US⟩ $ctxs = \emptyset$
4 **for each** $term \in usTerms$
5 **do** $ctxs$.add($userStories$.findStoriesThatContain($term$))
6 **for each** $(t1, t2) \in termPairs$
7 **do** $sim_{t1,t2}$ = semanticSiml($t1, t2$)
8 int $i = usTerms$.indexOf($t1$)
9 int $j = usTerms$.indexOf($t2$)
10 (Set⟨US⟩, Set⟨US⟩) $pairContext = (ctxs[i] \setminus ctxs[j], ctxs[j] \setminus ctxs[i])$
11 $simc_{t1,t2}$ = semanticSiml($pairContext$)
12 $ambig_{t1,t2} = \dfrac{2 \cdot sim_{t1,t2} + simc_{t1,t2}}{3}$

Illustration. Consider the following set of user stories: {us1 = As a t_A, I want ..., us2 = As a t_A, I want to print t_C ..., us3 = As a t_B, I want ..., us4 = As a t_A, I want to save t_C and t_B ..., us5 = As a t_B, I want to load t_C ...}. Visual Narrator (line 1) extracts the terms t_A, t_B, and t_C, while line 2 computes all pairs: (t_A,t_B), (t_A,t_C), and (t_B,t_C).

Lines 3–5 build the contexts for each term. For example, the context for t_A is {us1, us2, us4}, i.e., {As a t_A, I want ..., As a t_A, I want to print t_C ..., As a t_A, I want to save t_C and t_B ...}.

Lines 6–11 calculate the ambiguity score for each pair of terms. Take (t_A,t_B), and assume that Cortical.io returns a similarity score between the terms (line 7) of 0.34. The pair of contexts for those terms (line 10) is ({us1, us2}, {us3, us5}). The semantic similarity algorithm is now launched between the two elements of the pair of contexts; assume this results in a context similarity of 0.66 (line 11). Finally, the ambiguity score is determined in line 12 as $(2 \cdot 0.34 + 0.66)/3 = 0.44$.

3.1 Validation of the ambiguity score

We determined the weights for sim_p and sim_c based on the outcomes of exploratory tuning attempts: we have analyzed and discussed the outputs of dif-

ferent weights on training data sets and examples, and we found such weights to lead to results we perceived as the most representative for our data sets.

While robust, large-scale experiments are necessary to identify optimal values for the similarity values, we tested the reliability of $ambig_p$ with our weights via a correlation study between the algorithm and human judgment. The details on the experimental design and data are available online [18].

We employed the *WebCompany* data set that consists of 98 user story requirements. From this, taking the algorithm's outputs, we randomly extracted 8 term pairs with a high ambiguity score (≥ 0.6), 8 pairs with low ambiguity score (≤ 0.4), and 8 pairs with medium ambiguity score (between 0.4 and 0.6).

Eight master's students in information science participated voluntarily. Each of them filled in a questionnaire that contained 12 term pairs with their contexts (4 with low ambiguity, 4 medium, 4 high), with the terms allocated in such a way that every term pair would obtain the same number of judgments. For each term pair, the participant had to indicate how likely they perceived the term pair to be ambiguous, using the scale "Impossible", "Unlikely", "Likely", "Certain" or "Don't know". In total, 24 term pairs were processed by the 8 participants.

A Pearson correlation on the data shows a *strong* and *significant positive correlation* between the scores of the algorithm and by the participants, $r = .806$, $p = < .001$. Although the data is not sufficient to draw definite conclusions about generality and sensitivity, the results are promising.

4 Pinpointing Ambiguity and Incompleteness via InfoVis

Building on the framework of Table 1, we design a novel InfoVis technique for analysts to explore multiple viewpoints and for helping them pinpoint possible ambiguity and incompleteness. Our approach, also thanks to Algorithm 1, helps identify defects concerning the *correspondence* (synonyms and near-synonyms) and *contrast* relations (missing requirements). The *conflict relation* (homonyms) is supported to a more limited extent, as explained in this section.

Our visualization is inspired by our previous work on the automated extraction of conceptual models from user story requirements (the Visual Narrator tool) [8]. However, despite the high precision and recall, those models become quickly too large and models for humans to grasp and analyze. This is especially true when conducting in-depth analyses such as searching for defects.

To improve the situation, we resort to visualizing viewpoints via a Venn diagram, which is a suitable means for displaying overlapping elements [19]. Fig. 1 provides an example where the terms used from three viewpoints (by the stakeholders *Administrator*, *User* and *Visitor*) are shown alongside their overlap.

Finding (near-)synonymy. The visualization outlines the possibly ambiguous terms by applying Algorithm 1. A term's background color is set depending on the highest level of ambiguity that term possesses with respect to another term. As explained below (details-on-demand), this high-level overview can be refined for more accurate results.

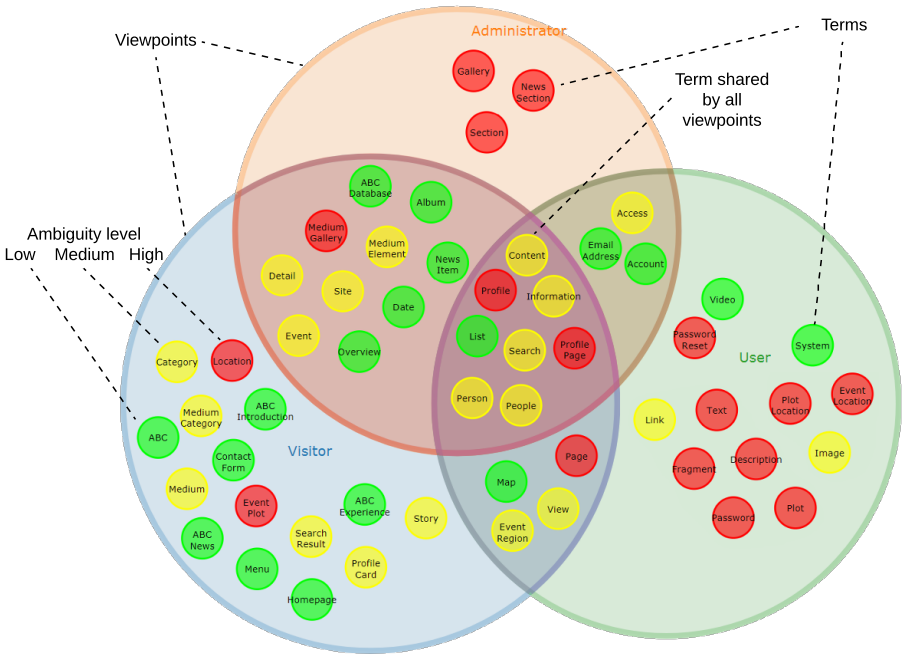


Fig. 1: Venn diagram visualization of three viewpoints and ambiguous terms.

Missing requirements and homonymy. Our approach helps an analyst explore the relationships between the terms used by multiple stakeholders. Consider the Venn diagram in Fig. 2 that includes three viewpoints, and whose intersection produces 7 areas (A–G)[2]. There are interesting areas for the analyst to examine:

- Areas A, C, G include terms that appear in a single viewpoint. These are loci where missing requirements may be discovered, because they contain terms that appear in a single viewpoint. In Fig. 1, for example, the term Plot appears only in the *User* viewpoint, but presumably also the *Administrator* may have some requirements about this content type.
- Area E contains the terms that are shared by all three viewpoints, while areas B, D, F include the terms that appear in exactly two viewpoints. The instances of every term therein—one or more instances per viewpoint—are either in consensus (no problem) or conflict (possible homonymy) relation. Determining which one of these two relations applies is up to the analyst, who should examine the user stories that contain those terms. This can be done using the details-on-demand zoom explained later in this section.

Filters. Our visualization comes with filters that can be applied to hide unwanted items from the display. We propose three filter types:

[2] Using triangular shapes, it is possible to show six viewpoints on a 2D space [20].

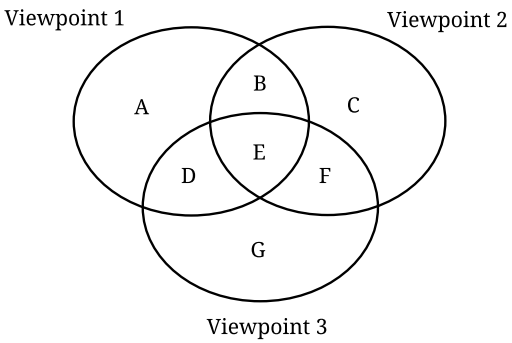


Fig. 2: The 7 areas (A–G) of our visualization applied to three viewpoints.

1. *Concept state filter* removes the concepts in a consensus/conflict state or those in a correspondence/contrast state from the display, so that the requirements engineer can focus on a given type of possible defects.
2. *Viewpoint filter* removes some viewpoints from the display, so that the analyst can focus on the remaining ones. This helps when more than three viewpoints exist; although it is possible to show six viewpoints without hiding any intersection [20], it is more practical to visualize two or three of them.
3. *Ambiguity filter* shows the elements within a given ambiguity score range. This can be useful to better examine the elements with high ambiguity score or to double check those with low-medium score. This is illustrated in Fig. 3.

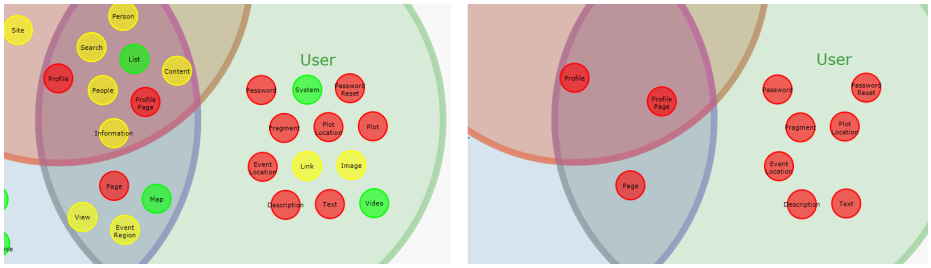
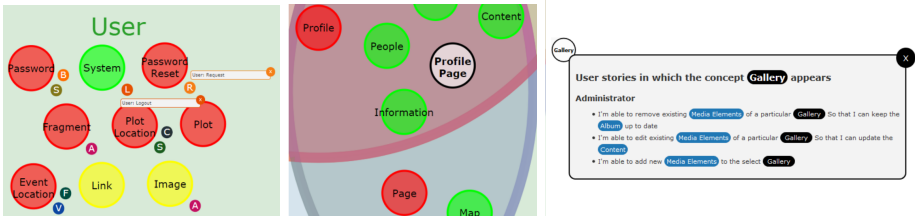


Fig. 3: Illustration of the ambiguity filter: on the right-hand side, only terms that are part of a term pair with an ambiguity score above 0.4 are shown.

Details-on-demand. These are features for retrieving additional details that are not visible through the main interface:

– *Association relationships* are the actions that a term refers to in the user stories. For example, in "As a user, I want to request a password reset", the



(a) Showing association relationships. (b) Ambiguity for term profile page. (c) User stories including term gallery.

Fig. 4: Illustration of details-on-demand.

association relationship of the term password reset is the verb request. When enabled, the association relationship is shown as a small icon next to the term. Each association relationship of a given term has a different color and is marked with the first character of the verb. Further details can be inspected by clicking on the icon, which opens a small pop-up window. Fig. 4a shows the association relationships for nine terms, and provides details for the verb request of term password reset, and for the verb logout of term system.

– *Ambiguity inspection.* The ambiguity that a term shares with other terms can be inspected by clicking on it. Boldface font is applied to the term label and the background is set to white, while the color of all other terms is changed based on the ambiguity score they share with the selected term. Fig. 4b shows high ambiguity between profile page and both profile and page.
– *User stories.* The user stories in which a term appears are shown in a pop-up window by double clicking on that term. The detailed term is given a black background, and other terms in those stories are given a blue background. Fig. 4c shows these details for the term gallery.

5 Evaluation

In Sec. 5.1, we show feasibility by describing our implementation of the approach presented in the previous sections. In Sec. 5.2, we report results from our preliminary evaluation of the tool effectiveness with groups of students.

5.1 Proof-of-Concept Tool

We developed a proof-of-concept Web 2.0 tool that implements the visualization described in Sec. 4 and the algorithm for ambiguity detection of Sec. 3. The tool is built on the Bootstrap framework, relies on the D3.js visualization library, and calls the REST API of cortical.io to compute semantic similarity.

The tool can be accessed online[3]. The website provides quick links to two sets of real-world user stories that showcase the tool's functionality: besides the *Web-Company* data set already mentioned, it is possible to explore the *CMS-Company*

[3] `http://www.staff.science.uu.nl/~dalpi001/revv/`

data set [8] that refers to a content management system. After importing the data sets, the viewpoints with the highest number of terms are shown.

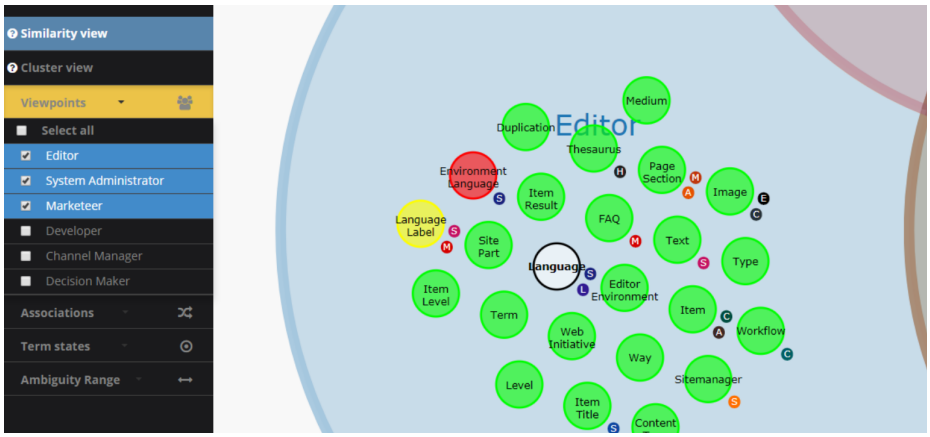


Fig. 5: Our tool showing an excerpt of the CMS-Company data set.

For example, in the CMS-Company data set, the three viewpoints shown by default are *Editor*, *System Administrator* and *Marketeer*, while the three less dense viewpoints are hidden: *Developer*, *Decision Maker* and *Channel Manager*. Fig. 5 shows an excerpt of the CMS-Company data set where the three main viewpoints are selected, and the analyst focuses on the term Language within the viewpoint *Editor*: the tool shows that Environment Language is likely to be a (near)-synonym of Language, while Language Label is less likely to be so.

The tool is a proof-of-concept. Although most functionalities are implemented, it is not a product. Also, the Venn-inspired visualization currently works with up to 3 viewpoints, while the functionality to support more than three concurrent viewpoints (through different shapes) has not been implemented yet.

5.2 Quasi-Experiment with Students

We report on a controlled quasi-experiment we conducted with students that aimed to assess the effectiveness of our approach as implemented by the tool described in Sec. 5.1. Our report follows Wohlin *et al.*'s guidelines [21].

Goal Definition and Context Selection. The goal of our evaluation as well as a description of the context selection are presented in Table 2.

Hypothesis Formulation. We derive four hypothesis by combining the two qualities we are interested in (precision and recall) with the two dependent variables: ambiguities and missing requirements. Therefore, our hypotheses unfold as follows: *Analysts who use our approach obtain a significantly higher X compared to analysts using a pen-and-paper inspection*, where X is as follows:

Table 2: Goal definition for our quasi-experiment.

Object of study	We study two objects: (i) Our tool-supported approach for identifying ambiguity and missing requirements supported by a wide 84" touch screen, and (ii) a manual, pen-on-paper inspection of the requirements.
Purpose	Evaluate the relative effectiveness of our approach compared to the pen-and-paper inspection.
Perspective	We take the point of view of RE researchers.
Quality focus	We study the *precision* and *recall* of the approach in detecting *ambiguity* and *incompleteness*.
Context	We involve voluntary master's students in Information Science from Utrecht University. We conduct a blocked subject-object study, for we have two objects and multiple subjects per object. Since we could not split the participants according to their background, we are conducting a quasi-experiment. The low number of students (n=8) makes the results preliminary.

- *precision in finding ambiguities* (H1);
- *recall in finding ambiguities* (H2);
- *precision in finding missing requirements* (H3);
- *recall in finding missing requirements* (H4);

Based on extensive brainstorming among the authors, a pilot test, and the existing literature, we have constructed the following pragmatic definitions of missing user stories and ambiguous user stories to use in our quasi-experiment; these definitions reflect the type of support that our tool intends to deliver:

- A missing user story is one whose absence inhibits the realization of at least another user story;
- An ambiguity occurs when two user stories contain distinct terms that shares the same denotations.

Experiment Design and Operation. We divided our 8 participants into four groups of two members each; two groups used our tool, while two groups used the pen-and-paper inspection. The participants had to work with a set of user stories that we assembled from a Software Architecture course, which they had attended; those user stories were for an event ticketing system.

The experiment was repeated two times; in each instance, we involved one group using our tool run on the interactive screen (*treatment* group) and one group performing manual inspection (*control* group). We executed three steps:

1. *Briefing (20 minutes)*: all participants read a 1-page document that described the experiment's goals (investigating the effectiveness of a visualization technique for finding ambiguities and missing user stories), included instructions, and provided an example. Then, the second author gave a short presentation about ambiguity and missing requirements. Finally, the members of the treatment group were given a 5-minute demo about our tool.
2. *Defect detection session (20 minutes)*: the two groups were assigned the task of finding ambiguities and missing user stories from the event ticketing system specification. They conducted their task in different rooms, with the second author unobtrusively observing the treatment group.

3. *Results evaluation (20 minutes)*: the groups collaborated toward identifying which of the identified ambiguities and missing requirements were true.

Validity Evaluation. We discuss the major threats to the validity of our study:

- *Internal validity.* The selection on participants based on their voluntary help made us unable to make a selection that evenly represents the entire population. While we tried to evenly balance the groups of participants based on our opinion on their skills and background knowledge, but we did not employ rigorous criteria to do so. Furthermore, relying on a discussion between group members to reach agreement on true ambiguities and missing requirements may suffer from social factors such as predominant personality and persuasion. Finally, the presence of an observer in the room with the treatment group may have affected the behavior exhibited by the participants.
- *Construct validity.* The pre-operational explication of constructs may have been unclear: ambiguity and incompleteness are difficult topics and, despite our attempt to use easily actionable definitions, the participants may have assigned different interpretations. Also, our treatment was influenced by a secondary factor, i.e., the use of an extra-large interactive screen, which could have affected the results. This threat is not extremely severe though, for this is the setting we designed our tool for. Furthermore, we did not study independently the effectiveness of the various features the tool embeds. Finally, it should be noted that the obtained results rely on the use of user story requirements; we cannot assess the generality for other notations.
- *Conclusion validity.* The small sample size implies low statistical power. Also, our study suffers from random heterogeneity of subjects, for it is likely that some individuals possessed significantly better analytical skills than others.
- *External validity.* The major threat in this category is that we chose students instead of professionals, for convenience reasons.

Analysis and Interpretation. The quantitative results of our quasi-experiment, obtained by running independent t-tests for H1–H4 based on the figures in Table 3, offer some interesting insights:

- *Precision (H1, H3)*: we cannot find any significant difference between the groups. For ambiguity, $t(-1.106) = -1.208$, $p = .442$, and Cohen's effect size value ($d = 1.1$) suggests low practical significance. For missing requirements, $t(-.044) = 1.283$, $p = .971$, and Cohen's effect size $d = 0.04$ suggest low practical significance.
- *Recall (H2, H4)*: we identify a significant difference in support of our hypotheses. For ambiguity, $t(-13.088) = 1.459$, $p = .017$, and $d = 13.2$, denoting high practical significance. For missing requirements, $t(-4.941) = 1.999$, $p = .039$, and $d = 4.999$, also suggesting high practical significance.

The results suggest to reject H1 and H3: our approach does not seem to increase precision in the identification of the stated defects. On the other hand, the results suggest to retain H2 and H4: our approach seems to lead to significantly higher recall compared to the pen-and-paper inspection. TThe validity of these results needs to be confirmed by replicating our study with more participants.

Table 3: Quantitative results of our quasi-experiment. TP and FP stand for true and false positives, respectively.

	Total TP	#TP	#FP	Precision	Recall
Session 1 – ambiguity					
Pen & paper	28	8	1	0.888	0.285
Tool		23	4	0.851	0.821
Session 2 – ambiguity					
Pen & paper	12	3	4	0.428	0.25
Tool		9	0	1	0.75
Session 1 – incompleteness					
Pen & paper	9	4	1	0.8	0.444
Tool		5	2	0.714	0.555
Session 2 – incompleteness					
Pen & paper	5	2	2	0.5	0.4
Tool		3	2	0.6	0.6

Qualitative results. We complement the results above with qualitative findings obtained from observation and via follow-up interviews with the participants:

- *Observations.* The two groups using the interactive screen behaved differently: while both members of the first group stood close to the screen and interacted with our tool, the second group had one person interacting with the screen and the other one standing at some distance to gain a more holistic viewpoint. The first group spent less time on identifying defects, perhaps due to the difficulty of obtaining a bird's eye view, and they also incurred in some conflicts, e.g., the tool interpreted as a pinch-to-zoom request the simultaneous drag-and-drop actions performed by the participants. In general, all the participants seemed confident with using either treatment even without an extensive training.
- *Interviews.* The participants provided suggestions that may help improve the tool. In particular, they identified some major missing features, such as a search field for quickly identifying the terms by name, a close-all button to hide all pop-up windows, and underlined association names in the details-on-demand window showing the user stories where a term appears in. Importantly, they indicated they would like to be able to change the elements in the visualization (rename and remove elements). They expressed appreciation for the tool, and found potential in terms of time saving thanks to the organization of the user stories around the viewpoints and the terms that occur in the stories.

6 Related work

InfoVis for RE. A recent systematic literature review [22] classifies existing approaches along the RE activities they support, the involved stakeholders, and

the focus on the problem or solution domain. According to that framework, our work supports the *requirements verification* activity, focuses on the *problem domain* (stakeholders' needs), and is intended for *decision-makers*.

Among existing visualization approaches, a similar approach to ours is taken by Savio *et al.* [23], who propose a 3D pyramidal visualization in which every face of the pyramid represents one stakeholder/viewpoint, and the pyramid is sliced along the z-axis to denote different levels of refinement of the requirements. Reddivari *et al.* [24]'s RecVisu+ tool organizes requirements graphically in clusters based on their similarity, it includes an algorithm for automated cluster label generation, and it supports manipulating the requirements during their elaboration. Orthogonally, the atomic elements in our approach are the terms (instead of the requirements), and the analyst can then inspect the corresponding requirements by requesting details (see Fig. 4c).

In our previous work [25], we proposed a cluster-based visualization of the terms extracted from user story requirements. Differently, in this paper the terms are not aggregated via clustering, but they are organized according to viewpoints, and ambiguity detection algorithms support the identification of possible defects.

Ambiguity in RE. Several studies on ambiguity in RE have been conducted so far. The seminal contribution of Berry and Kamsties [1] provides an excellent overview of the main categories of ambiguity and their relevant for RE, including lexical (investigated in this paper), syntactic or structural, semantic, and pragmatic. Since then, researchers have examined anaphoric ambiguity (pronouns) [26]; proposed dictionary-based approaches to detect ambiguous and weak terms [27]; introduced the notion of nocuous ambiguity [28] as opposed to harmless ambiguity; experimented what combinations of ambiguity metrics is more effective in practice [29]; and studied pragmatic ambiguity that depends on the background of the reader [30]. Our work adds another brick to this thread of research, as our techniques focus on pinpointing near-synonymy and homonymy.

7 Discussion

We have proposed an approach that combines InfoVis and NLP in order to help analysts identify some classes of ambiguity (near-synonymy and homonymy) and missing requirements. Our visualization represents the requirements graphically by highlighting the terms that are used and arranges those terms on a 2D space according to the viewpoint they belong to. Our preliminary evaluation suggests that our approach *may* lead to a significantly better recall than a pen & paper inspection, while no significant difference in precision could be detected.

Several research challenges need to be overcome. The effectiveness should be tested at a larger scale and possibly by isolating the effect of the individual tool functionalities. The algorithm for detecting ambiguity can be improved and tuned, while avoiding over-fitting. We also wish to study whether domain ontologies can lead to a deeper understanding of the requirements and their relationships. We would like to identify visualization mechanisms that avoid heavy

reliance on colors, which are an obstacle for color-blinded people. Finally, we are interested in the use of InfoVis techniques to ease the transition from RE to architectural design.

More generally, this paper opens the doors for future work that combines InfoVis and NLP. While we examined incompleteness and ambiguity in user stories, other requirements notations and other defect types should be studied.

Acknowledgment

This project has received funding from the SESAR Joint Undertaking under grant agreement No 699306 under European Union's Horizon 2020 research and innovation programme.

References

1. Berry, D.M., Kamsties, E., Krieger, M.M.: From contract drafting to software specification: Linguistic sources of ambiguity. Technical report, School of Computer Science, University of Waterloo, Canada (2001)
2. Bano, M.: Addressing the challenges of requirements ambiguity: A review of empirical literature. In: Proc. of the International Workshop on Empirical Requirements Engineering. (2015) 21–24
3. Rosadini, B., Ferrari, A., Gori, G., Fantechi, A., Gnesi, S., Trotta, I., Bacherini, S.: Using NLP to detect requirements defects: An industrial experience in the railway domain. In: Proc. of the International Working Conference on Requirements Engineering: Foundation for Software Quality. (2017) 344–360
4. Cambria, E., White, B.: Jumping NLP curves: A review of natural language processing research. IEEE Computational Intelligence Magazine **9**(2) (2014) 48–57
5. Berry, D., Gacitua, R., Sawyer, P., Tjong, S.: The case for dumb requirements engineering tools. In: Proc. of the International Working Conference on Requirements Engineering: Foundation for Software Quality. Volume 7195 of LNCS. (2012) 211–217
6. Lucassen, G., Dalpiaz, F., van der Werf, J.M.E.M., Brinkkemper, S.: Improving agile requirements: The Quality User Story framework and tool. Requirements Engineering **21**(3) (2016) 383–403
7. Cohn, M.: User Stories Applied: for Agile Software Development. Addison Wesley Professional, Redwood City, CA, USA (2004)
8. Robeer, M., Lucassen, G., Van der Werf, J.M., Dalpiaz, F., Brinkkemper, S.: Automated extraction of conceptual models from user stories via NLP. In: Proc. of the International Requirements Engineering Conference. (2016)
9. Finkelstein, A., Kramer, J., Nuseibeh, B., Finkelstein, L., Goedicke, M.: Viewpoints: A framework for integrating multiple perspectives in system development. International Journal of Software Engineering and Knowledge Engineering **2**(1) (1992) 31–57
10. Mullery, G.P.: CORE - a method for controlled requirement specification. Proc. of the International Conference on Software Engineering (1979) 126–135
11. Sommerville, I., Sawyer, P.: Viewpoints: principles, problems and a practical approach to requirements engineering. Annals of Software Engineering **3**(1) (1997) 101–130

12. Shaw, M.L., Gaines, B.R.: Comparing conceptual structures: consensus, conflict, correspondence and contrast. Knowledge Acquisition **1**(4) (1989) 341–363
13. Pohl, K.: Requirements Engineering: Fundamentals, principles, and techniques. Springer (2010)
14. DiMarco, C., Hirst, G., Stede, M.: The semantic and stylistic differentiation of synonyms and near-synonyms. In: Proc. of the AAAI Spring Symposium. (1993) 114–121
15. Rips, L.J., Shoben, E.J., Smith, E.E.: Semantic distance and the verification of semantic relations. Journal of Verbal Learning and Verbal Behavior **12**(1) (1973) 1–20
16. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Proc. of the Neural Information Processing Systems Conference. (2013) 3111–3119
17. De Sousa Webber, F.: Semantic folding theory and its application in semantic fingerprinting. arXiv preprint arXiv:1511.08855 (2015)
18. van der Schalk, I.: REVV: A tool to create a better understanding of software requirements through Information Visualization and NLP. Master's thesis, Utrecht University (2017)
19. Micallef, L.: Visualizing set relations and cardinalities using Venn and Euler diagrams. PhD thesis, University of Kent (2013)
20. Carroll, J.J.: Drawing Venn triangles. HP Labs Technical Report (73) (2000)
21. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A.: Experimentation in software engineering. Springer Science & Business Media (2012)
22. Abad, Z.S.H., Ruhe, G., Noaeen, M.: Requirements Engineering visualization: A systematic literature review. In: Proc. of the International Requirements Engineering Conference. (2016)
23. Savio, D., Anitha, P., Patil, A., Creighton, O.: Visualizing requirements in distributed system development. In: Proc. of the Workshop on Requirements Engineering for Systems, Services and Systems-of-Systems, IEEE (2012) 14–19
24. Reddivari, S., Rad, S., Bhowmik, T., Cain, N., Niu, N.: Visual requirements analytics: A framework and case study. Requirements Engineering **19**(3) (2014) 257–279
25. Lucassen, G., Dalpiaz, F., van der Werf, J.M.E., Brinkkemper, S.: Visualizing user story requirements at multiple granularity levels via semantic relatedness. In: Proc. of the International Conference on Conceptual Modelling. (2016) 463–478
26. Yang, H., De Roeck, A., Gervasi, V., Willis, A., Nuseibeh, B.: Analysing anaphoric ambiguity in natural language requirements. Requirements Engineering **16**(3) (2011) 163
27. Tjong, S.F., Berry, D.M.: The design of SREE: A prototype potential ambiguity finder for requirements specifications and lessons learned. In: Proc. of the International Working Conference on Requirements Engineering: Foundation for Software Quality. Volume 7830. (2013) 80–95
28. Willis, A., Chantree, F., De Roeck, A.: Automatic identification of nocuous ambiguity. Research on Language & Computation **6**(3) (2008) 355–374
29. Kiyavitskaya, N., Zeni, N., Mich, L., Berry, D.M.: Requirements for tools for ambiguity identification and measurement in natural language requirements specifications. Requirements Egineering **13**(3) (2008) 207–239
30. Ferrari, A., Lipari, G., Gnesi, S., Spagnolo, G.O.: Pragmatic ambiguity detection in natural language requirements. In: Proc. of the International Workshop on Artificial Intelligence for Requirements Engineering, IEEE (2014) 1–8