Hacking an Ambiguity Detection Tool to Extract Variation Points: an Experience Report

Alessandro Fantechi
Dipartimento di Ingegneria dell'Informazione,
Università di Firenze
Firenze, Italy
alessandro.fantechi@unifi.it

Alessio Ferrari
Istituto di Scienza e Tecnologie dell'Informazione
"A.Faedo", Consiglio Nazionale delle Ricerche, ISTI-CNR
Pisa, Italy
alessio.ferrari@isti.cnr.it

Stefania Gnesi
Istituto di Scienza e Tecnologie dell'Informazione
"A.Faedo", Consiglio Nazionale delle Ricerche, ISTI-CNR
Pisa, Italy
stefania.gnesi@isti.cnr.it

Laura Semini
Dipartimento di Informatica, Università di Pisa
Pisa, Italy
semini@di.unipi.it

ABSTRACT

Natural language (NL) requirements documents can be a precious source to identify variability information. This information can be later used to define feature models from which different systems can be instantiated. In this paper, we are interested in validating the approach we have recently proposed to extract variability issues from the ambiguity defects found in NL requirement documents. To this end, we single out ambiguities using an available NL analysis tool, QuARS, and we classify the ambiguities returned by the tool by distinguishing among false positives, real ambiguities, and variation points.

We consider three medium sized requirement documents from different domains, namely, train control, social web, home automation. We report in this paper the results of the assessment. Although the validation set is not so large, the results obtained are quite uniform and permit to draw some interesting conclusions.

Starting from the results obtained, we can foresee the tailoring of a NL analysis tool for extracting variability from NL requirement documents.

KEYWORDS

NLP, natural language, requirements, variability, ambiguity.

1 INTRODUCTION

The identification of variability in different system artifacts, such as requirements, architecture and test cases, is one of the cornerstone activities of software product line engineering (SPLE) [9]. Several methods were developed for variability identification and management that are specifically focused on *requirements*, including feature-oriented domain analysis (FODA) [27], the RequiLine tool [39], the domain requirements model (DRM) based approach [34], as well as the work of Moon *et al.* [30]. In recent years, with the increasing capabilities of natural language processing (NLP) tools [23], a trend has emerged in variability identification methods, which is based on extracting features and variability-related information from natural language (NL) documents in general [16, 29, 31] and requirements in particular [4, 26].

In line with this stream of research, in [12] we have discussed how to extract variability issues from a requirements document using NLP tools. In particular we have focused on using tools aimed at revealing the *ambiguity* defects of the NL sentences in the requirements document. The underlying intuition is that often ambiguity in requirements is due to the (conscious or subconscious) need to postpone choices for later decisions in the implementation of the system. Ambiguity in NL has been largely studied in requirements engineering (RE), and several approaches have been developed to automatically detect defective expressions that can be interpreted in different ways by the stakeholders who have to read the requirements [6, 13, 21, 35, 38]. These approaches focus on identifying typically vague terms, such as adjective and adverbs (e.g., [21, 38]), and ambiguous syntactic construction due to the use of pronouns [42], or coordinating conjunctions such as "and" or "or" [7, 43]. Our work differs from these ones, in that ambiguity is not regarded as a defect, but it becomes a means to enlighten possible variation points in an early phase of software and system development, and give space for a range of different products. This view stems from the observations in [18], in which ambiguity is considered a resource, rather than an obstacle, to disclose implicit information (i.e., *tacit knowledge* [20]).

In our approach [12], ambiguities are first identified by means of an automated NLP tool. Then, a requirements analyst reviews the output of the tool, and identifies which of the identified ambiguities can be considered as: (a) true NL ambiguities, (b) innocuous ambiguities, i.e. false positive cases, and (c) variation points.

In this paper, we show the first results achieved with experimenting this approach by means of the QuARS tool for ambiguity detection [22], applied on three different requirement documents. The documents belong to different domains, and consist of 572 requirements in total. The QuARS tool is able to point to ambiguity defects in the documents according to different indicators; we then evaluate the outcome of the tool to rate the relevance of the found defects to express variability, as well as the ability of the tool to detect, by means of ambiguity detection, all the variability present in the requirements. Our results confirm that a relevant part of the ambiguity detected by QuARS can be considered as source of variation points. The analysis of the cases offer hints to identify potential ways to tailor the QuARS tool to specifically support variability identification.

The remainder of the paper is structured as follows. In Sect. 2 we present related works. In Sect. 3 we provide background on our study and on the QuARS tool. Sect. 4 describes the research questions, the overall design of our study and the requirement documents that we used for the study. In Sect. 5 we discuss the results produced by the analysis run with QuARS on the selected documents, and in Sect. 6 we discuss the limitations of the current work. Sect. 7 concludes the paper and provides final remarks.

2 RELATED WORK

The work presented in this paper is concerned with feature identification from NL documents, and with ambiguity detection in NL requirements. Therefore, we briefly discuss the related work in these two fields.

Feature Identification from NL Documents. The works that are concerned with automated feature identification from NL texts can be classified into works that focus on requirements, and works that leverage other types of system-related documents, and, more specifically, product descriptions.

Among the works that focus on requirements, the DARE tool [19] is the earliest contribution. A semi-automated approach is employed to identify features according to lexical analysis based on term frequency (i.e., frequently used terms are considered more relevant for the domain). Chen *et al.* [8] suggests the usage of the *clustering* technology to identify features: requirements are grouped together according to their similarity, and each group of requirements represents a feature. Clustering is also employed in subsequent works [2, 32, 33, 40], but while in [8] the computation of the similarity among requirements is *manual*, in the other works *automated* approaches are employed. Among the relevant works, Weston *et al.* [40] used Latent Semantic Analysis (LSA) to extract the so-called Early Aspects. These are cross-cutting concerns that are useful to derive features. Niu *et al.* [32, 33] use Lexical Affinities (LA) – roughly, term co-occurrences – as the basis to find representative expressions (named Functional Requirements Profiles) in functional requirements. Finally, Itzik *et al.*[26] presents the SOVA approach, which leverages semantic ontologies to extract features from requirements and derive a feature model.

Other works [1, 10, 16, 31] present approaches where *product descriptions* are used. The feature mining methodology presented by Dimitru *et al.* [10] is based on clustering, and the authors provide also automated approaches to recommend useful features for new products. Instead, the approach presented by Acher *et al.* [1] is based on searching for variability patterns within tables in which the description of the products are stored in a semi-structured manner. Ferrari *et al.* [16] extract domain-specific terms from product descriptions belonging to different vendors, to identify common and variant domain terms, which can be used as pointers for product commonalities and variabilities. Nasr *et al.* [31] leverage an approach analogous to the one used in [16], to derive comparison matrices for different products.

Systematic literature reviews on feature extraction and variability extraction from NL documents have been published by Li *et al.* [29] and by Bakar *et al.* [4].

Ambiguity Detection in NL Requirements. Ambiguity detection in requirements is a lively research field, with several contributions published already in the nineties (e.g., the ARM tool [41]), and recent industrial applications [13, 35]. Most of the works stem from the typically defective terms and constructions classified in the ambiguity handbook of Berry *et al.* [24]. Based on these studies, rule-based NLP tools such as QuARS [22], SREE [38] and the tool of Gleich *et al.* [21] were developed. More recently, industrial applications of these approaches were studied by Femmer *et al.* [13] and by Rosadini *et al.* [35]. Furthermore, Arora *et al.* [3] presented RETA (REquirements Template Analyzer), a tool that employs rule-based approaches to detect typical ambiguous terms and constructions, as the other mentioned works, and, in addition, checks the conformance of the requirements to a given template.

As shown also in these studies, rule-based approaches tend to produce a high number of false positive cases – i.e., linguistic ambiguities that have one single reading in practice. Hence, *statistical* approaches were proposed by Chantree *et al.* [7] and by Yang *et al.* [42] to reduce the number of false positive cases, referred as *innocuous ambiguities*. Statistical NLP approaches are also used in [15], to identify domain-dependent ambiguities, i.e., pragmatic ambiguities that depend on the domain background of the reader of the requirements.

Our work differs from the contributions in the two fields, in that it integrates the research in ambiguity detection, with the research in feature identification. More specifically, we *hack* the ambiguity detection capabilities of the QuARS tool to identify variation points in requirements documents. The closest works in feature identification are those that focus on variant feature identification from NL documents, as, e.g., [16, 31]. However, these works leverage the automated extraction of domain-specific terms, while in this work we focus on ambiguity detection.

3 BACKGROUND

Requirements are an abstract description of the system needs that is inherently open to different interpretations [14]. This openness is emphasized by the use of NL, which is intrinsically ambiguous, even though it is commonly used to express requirements [28]. Indeed, NL is the most widely used communication code, since it easily supports the exchange of knowledge among different stakeholders

Sub-characteristic	*Indicators*
Vagueness	The occurrence of Vagueness-revealing wordings (such as e.g.: clear, easy, strong, good, bad, useful, significant, adequate, recent, ...) is considered a vagueness indicator
Subjectivity	The occurrence of Subjectivity-revealing wordings (such as e.g.: similar, similarly, having in mind, take into account, as [adjective] as possible, ...) is considered a subjectivity indicator
Optionality	The occurrence of Optionality-revealing words (such as e.g.: possibly, eventually, case, if possible, if appropriate, if needed, ...) is considered an optionality indicator
Implicity	The occurrence of: • Subjects or complements expressed by means of: Demonstrative adjectives (this, these, that, those) or Pronouns (it, they...) or • Terms having the determiner expressed by a demonstrative adjective (this, these, that, those) or implicit adjective (such as e.g. :previous, next, following, last...) or preposition (such as e.g.: above, below...) is considered an implicity indicator
Weakness	The occurrence of Weak verbs (such as e.g.: may) is considered a weakness indicator
Under-specification	The occurrence of words needing to be instantiated (such as e.g.: information, interface, that must be better defined, flow instead of data flow, control flow, access instead of write access, remote access, authorized access, testing instead of functional testing, structural testing, unit testing, etc.) is considered an under-specification indicator.
Multiplicity	The occurrence of multiplicity-revealing words: and, and/or, or, ... is considered a multiplicity indicator.

Table 1: QuARS ambiguity indicators

with heterogeneous backgrounds and skills. As the requirements process progresses, requirements are expected to be sufficiently clear to be interpreted in an unequivocal way by the interested stakeholders [14].

A solution found within the RE community is to employ NLP tools that make the editors aware of the ambiguity in their requirements [22, 38]. Ambiguities normally cause inconsistencies between the expectation of the customer and the product developed, and possibly lead to undesirable reworks on the artifacts. However, ambiguity can also be used as a way to capture variability aspects to be solved later in the software development.

In [12] we proposed a first classification of the forms of ambiguity that indicate variation points, and we described a possible mapping from ambiguity indicators to fragments of feature models. Specifically, we envisioned an approach to achieve automated support to variability elicitation by analysing the outcomes of automated ambiguity detection applied to some set of requirements by means of the QuARS (Quality Analyser for Requirements Specifications) tool [11, 22], one of the leading tools addressing NLP of requirement documents. In the current paper, the approach, preliminarily defined in [12], is systematically assessed on a dataset of 572 requirements, coming from three different documents.

3.1 QuARS

QuARS was introduced as an automatic analyzer of requirement documents [22]. QuARS performs an initial parsing of NL requirements for automatic detection of potential linguistic defects that can determine ambiguity problems impacting the following development stages. QuARS performs a linguistic analysis of a requirements document in plain text format and points out the sentences that are defective according to the expressiveness quality model described in [5]. The defect identification process is split in two parts: (i) the "lexical analysis" capturing *optionality, subjectivity, vagueness, multiplicity and weakness* defects, by identifying candidate defective words that are identified into a corresponding set of dictionaries; and (ii) the "syntactical analysis" capturing *implicitness and under-specification* defects. In the same way, detected defects may however be *false defects.* In Table 1 we present the indicators used by QuARS to detect lexical and syntactical defects in NL sentences.

Other functionalities, not related to the aim of this paper, are offered by QuARS, like requirements clustering, metrics derivation for evaluating the quality of NL requirements and view derivation, to identify and collect together those requirements belonging to given functional and non functional characteristics.

4 RESEARCH METHODOLOGY AND STUDY DESIGN

In the experience reported in this paper, QuARS is used to point to ambiguity defects in a sample of requirements documents. For this purpose, three requirements documents have been chosen, coming from three different domains. The three documents are scanned by QuARS, and the reported defects are then analysed by a human expert to see whether they point to a possible variability of the described system: that is, each defect is analysed to judge whether it is not a defect, but rather points to different choices that can give space for a range of different products.

4.1 Research Objective and Research Questions

The objective of this study is to assess whether ambiguities in NL requirements can be considered as potential variation points, and to which extent the process of variability identification can be automated with an ambiguity detection tool such as QuARS. This objective is decomposed into the following research questions (RQs):

RQ1 Is automated ambiguity detection in NL requirement documents relevant to detect variability?

This question can be answered by giving measures about how many variabilities are identified out of the total ambiguities detected by QuARS, and how many are instead false positives.

RQ2 Are all of the ambiguity indicators relevant or only some of them?

This question is oriented to understand which, among the indicators provided by QuARS, are the most relevant to detect variation points. The underlying goal is to identify whether QuARS can be tailored to detect variation points by focusing solely on a specific subset of the provided indicators.

RQ3 Can we derive from this assessment new terms and parameters for tuning existing NL requirements analysis tools?

This question can be answered by inspecting false positive cases produced by QuARS, and by understanding which of the cases can be systematically detected, so that the capabilities of the tool for variability identification can be improved.

RQ4 To which extent is automated ambiguity detection in NL requirement documents a complete instrument to detect variability?

This question can be answered by giving measures about how many variabilities that are actually present in the requirement document, as identified by expert judgement, are not identified as ambiguity defects by QuARS (i.e., false negatives). In this paper, a partial answer to this question will be provided, since only a non-systematic inspection is performed to check false negative cases. A complete answer to RQ4 requires to annotate variation points in the documents before QuARS is executed, and then to inspect the false negatives in a systematic manner. Given the exploratory nature of the current study, this activity is left as future work.

4.2 Case Selection and Description

We base our experience on three requirements documents very different from each other: different domains, different characteristics of the systems, different background and experience of of their authors. The three documents are briefly described below. The first and third document can be downloaded from the PURE requirements dataset described by Ferrari *et al.* [17], and available at the following link: http://fmt.isti.cnr.it/nlreqdataset/ (file names: 2007 - ertms, 2010 - home 1.3). The second document is available at the following link: https://www.plat-forms.org/sites/plat-forms.org/files/platforms-task.pdf.

4.2.1 ERTMS: train control system. The first document we have considered defines the functional requirements for ERTMS/ETCS (European Rail Traffic Management System / European Train Control System), issued by the European Railway Agency in June 2007. The document includes 96 requirements of a control system that supports the driver of a train: it provides the driver with information needed for the safe driving of the train, and it is able to supervise train and shunting movements.

4.2.2 People by Temperament: social web application. Our second document comes from Plat_Forms, an international academic-industrial programming contest. It aims at comparing different technological platforms for developing web-based applications. We have chosen the requirements given at the first edition of the contest, in 2007. The system to be built is called PbT (People by Temperament), a simple community portal where members can find others with whom they might like to get in contact: people register to become members, take a personality test, and then search for others based on criteria such as personality types, likes/dislikes etc. The documents includes 325 requirements.

4.2.3 DigitalHome: home automation system. This document specifies the requirements for the development of a *Smart House*, called DigitalHome (DH). The DH case study material has been developed and used as a case study throughout a computing curriculum [25], as part of a US National Science Foundation grant. The DH system allows a home resident to manage devices that control the environment of a home. The user communicates through a web page on a web server. The DH web server communicates, through a wireless gateway device, with the sensor and controller devices in the home.

The document was developed by a team of 5 students in an academic context, and includes 151 requirements.

4.3 Data Collection and Analysis

In [12], we have presented the idea that under-specification or ambiguity at requirements level can in some cases give an indication of possible variability, either in design choice, in implementation choices or configurability. Taking into account the results of previous analyses conducted on different requirements documents with NL analysis tools, we attempted a first classification of the forms of ambiguity that indicate variation points, and we indicated an approach to achieve automated support to variability elicitation.

We now address the validation of this idea, by first analysing, using QuARS, the three requirement documents described in Sect. 4.2 according to all the indicators given in Table 1.

Then, to elicit the potential variability hidden in a requirement document, we perform an assessment of the output of the tool, for each ambiguity indicator, aimed at classifying the defective sentences and distinguish among: false positives, variability points, and actual ambiguities.

More specifically, the *data collection* procedure, for each document, consists of the following steps:

(1) **Automatic Detection:** The document is given as input to QuARS in textual format, and QuARS produces a set of sentences that are considered ambiguous, together with the term or expression that is the source of the ambiguity;
(2) **Review:** The output of QuARS is reviewed by the 4th author, who classifies each defect identified by QuARS as false positives, variability indicator, or actual ambiguity;
(3) **Assessment:** The classification is reviewed by the 3rd author, and, if discrepancies emerge in the judgment, agreement is reached through discussion.

Review and assessment phases, that highlight variation points, are based on the criteria introduced in our previous paper [12]. We recall here the main ideas. Ambiguity in requirements may be due to the need to enlighten possible variation points in an early phase of software and system development and to postpone choices for later decisions in the implementation of the system. Hence, the analysis of the defective requirements is guided by the general question "Can this requirement hide a variation point?". More concrete criteria depend on the indicators. In the cases of *implicity* and *subjectivity*, there is no intuition that a defect can actually be a variation point, and the analysis is performed in a completely subjective way. With *under-specification* and *vagueness* the criterium is the existence of more than one possible instance of the defective word. With *multiplicity* the assessor can discard all requirements where conjunction/disjunction relate two sentences or two adjective, and concentrate on the cases where they relate nouns. The cases of *weakness* and *optionality* are treated similarly, since the nature of these defects is inherently associated to variation points, especially when they appear in functional requirements. Subjective judgment is adopted in case of non-functional requirements.

The *data analysis* procedure, for each document, consists of the following steps:

(1) **Quantitative Analysis:** The number of defects found by the tool (FND), false positives (FP), variability indicators (VAR), and the actual ambiguities (AMB) is computed for each indicator. This evaluation aims at answering RQ1, and to give a broad view about the indicators that are more relevant for variability detection (RQ2).
(2) **Qualitative Analysis:** For each indicators, typical classes of variability-related terms are identified, as well as typical cases of false positive. This analysis aims to provide a more informed answer to RQ2, and to answer RQ3. Furthermore, a non-systematic inspection is performed on the original requirements, to check whether certain classes of false negatives could be identified, in order to provide a preliminary answer to RQ4.

5 RESULTS

Tables 2 to 7 show the results of the quantitative analysis: each table addresses one of the six QuARS indicators, as computed for each case. In each table and per each case study, in the first column we report the number of defects found by the tool (FND), and in the next columns the number of false positives (FP), the variability indicators (VAR), and the actual ambiguities (AMB), as classified by manual inspection.

Let us comment each of the tables. For the *implicity* indicator, a sentence is considered defective if its subject or complements are *implicit*, being expressed by demonstrative adjectives (this, these, that, those) or pronouns (it, they, *etc.*) instead of by a noun. Table 2 tells that in the considered documents, implicity is in most cases resolved when reading the sentence, and, in any case, it is never an indication of possible variability. A requirement that can be considered as ambiguous is for instance: *TakeTtt [...] evaluates one set of answers to the TTT, computes the TTT result and TTT type, and stores* <u>*them*</u> *(plus a timestamp) for the current user.*

	IMPLICITY			
	FND	FP	VAR	AMB
ERTMS	2	2	0	0
DigitalHome	9	9	0	0
People by Temperament	21	18	0	3
Total	32	29	0	3

Table 2: Classification of implicity defects

Also in the case of *under-specification*, most defective sentences are false positives, and almost no variability is hidden behind (Table 3). Only in ERTMS, the word *information* is a variability candidate: it appears twice in a sentence of the kind *ETCS shall provide the driver with* <u>*information*</u> *to allow him/her to safely drive the train.* The amount of <u>information</u> provided to the driver can vary and indeed can be configured differently in different countries or for different typologies of rolling stock. On the contrary, the term *information* is considered an ambiguity in *The user documentation shall include the following: [...] A section that explains how DH parameters are set and sensor values are read. This shall include* <u>*information*</u> *on limitations and constraints on parameter settings and sensor reading accuracy.*

	UNDER-SPECIFICATION			
	FND	FP	VAR	AMB
ERTMS	6	4	2	0
DigitalHome	15	14	0	1
People by Temperament	2	2	0	0
Total	23	20	2	1

Table 3: Classification of under-specification defects

With *multiplicity*, variability is actually an option when disambiguating (see Table 4) and in most cases false positives (RQ3) are due to sentences with two verbs. The following requirement of DigitalHome exemplifies this affirmation, since it contains a variability point ("or") and a false positive ("and"): *The DigitalHome programmable thermostat shall allow a user to monitor* <u>*and*</u> *control a home's temperature from any location, using a web ready computer, cell phone,* <u>*or*</u> *PDA*. These cases can be potentially discarded by employing POS Tagging [37] – i.e., identification of verbs, nouns, conjunctions, *etc.* – and by identifying all the cases in which the term "and" occurs between two verbs[1].

Other systematic false positive cases for *multiplicity* occur when coordinating conjunctions are used between values to indicate a range. For example, consider the case: *The sensor part of the thermostat has a sensitivity range between* 14^oF <u>*and*</u> 104^oF. These cases can be automatically discarded by defining NLP patterns that recognise, e.g., occurrences of coordinating conjunctions between numerical amounts, possibly associated to units of measurement.

A requirement that includes a multiplicity indicator, but cannot be considered as a case of variability is: *The life motto is an arbitrary one-line phrase* <u>*or*</u> *sentence meant to characterize the person.*

[1]In the cases in which an adverb is attached to the two verbs, an attachment ambiguity [24] may occur. Hence, these cases may require specific treatments.

	MULTIPLICITY			
	FND	FP	VAR	AMB
ERTMS	30	24	6	0
DigitalHome	137	80	46	11
People by Temperament	125	80	18	27
Total	292	184	70	38

Table 4: Classification of multiplicity defects

Table 5 reports the results for *subjectivity*: at least for these case studies this indicator is not relevant. To decide if this observation scales, we need to examine a larger set of case studies.

	SUBJECTIVITY			
	FND	FP	VAR	AMB
ERTMS	0	0	0	0
DigitalHome	0	0	0	0
People by Temperament	5	5	0	0
Total	5	5	0	0

Table 5: Classification of subjectivity defects

Vagueness is due to the presence of undetermined adjectives and adverbs and, as reported in Table 6, can mask a variability. An example is the following: *The user interface should provide sufficient explanation of all uncommon concepts to guide the user.* Indeed, the detail level of the user interface can vary in different products. Another example is: *The general user shall be able to use the DH system capabilities to monitor and control the environment for his/her home.* In this case the term *general* may indicate that more than one type of user is foreseen for the system (i.e., the Digital Home (DH), in this case).

Typical false positives (RQ3) for vagueness are those in which a certain term is *systematically polysemous* [24], and it is used in the form of noun, instead of, e.g., adjective. Examples include the term *light* and *sound*, as in the following requirement: *The system shall include security sound and light alarms.* These cases can be discarded by including POS Tagging, and identifying when certain vague terms are used in the form of nouns, as performed by Rosadini *et al.* [35].

A requirement with three ambiguities is: *[...] such failures might affect the safety of home dwellers (e.g., security breaches, inadequate lighting in dark spaces, inappropriate temperature and humidity for people who are in ill-health, or powering certain appliances when young children are present).*

	VAGUENESS			
	FND	FP	VAR	AMB
ERTMS	2	2	0	0
DigitalHome	35	24	7	4
People by Temperament	39	34	2	3
Total	76	60	9	7

Table 6: Classification of vagueness defects

A further ambiguity indicator of QuARS is *weakness*: a sentence with verb *may* is considered weak. Besides, sometimes requirements are labelled with a *may* to indicate that their implementation is optional and introducing a variability. A good percentage of the defective sentences revealed by QuARS express optional requirements, as shown in Table 7. A couple of typical examples follow: *Clicking on the symbol of a member in the graphic may call that member's Status Page; The portal may work fully with other browsers such as Konqueror, Opera Mini, Lynx etc.* On the contrary, an ambiguity is in: *The system shall include security sound and light alarms, which can be activated when DigitalHome senses a security breach from a magnetic contact.*

	WEAKNESS			
	FND	FP	VAR	AMB
ERTMS	4	0	4	0
DigitalHome	10	4	1	5
People by Temperament	47	12	31	4
Total	61	16	36	9

Table 7: Classification of weakness defects

The last ambiguity indicator of QuARS is *optionality*, revealed by expressions like *if possible, if needed* etc. We did not find any optionality defect in any of the considered documents.

The data reported in the Tables 2-7 and the above discussion of the quantitative analysis shows some answers to our research questions: indeed the use of an ambiguity detection tool for NL requirements can be helpful to detect variability (RQ1) and only some of the ambiguity indicators are significant, namely: multiplicity, vagueness, and weakness (RQ2). Hence, a NL analysis tool can be restricted to consider only these indicators when used to elicit variability, and this partly answers RQ3. Furthermore, the systematic false positive cases identified for multiplicity and vagueness offer further hints to tailor QuARS in order to increase its accuracy in terms of variability identification (RQ3).

Another issue is to look for the false negatives, in order to answer RQ4, and provides further hints to improve the tool (RQ3). How many variability points can be found in the requirement documents, which were not found by QuARS? Do they respect some rule, so that an automatic tool can be instructed to detect them? We only have a partial answer here, for the more frequent cases, given the non-systematic inspection activity performed at this stage. Two general cases of false negatives were identified. (1) All the occurrences of a list can correspond to an *and* multiplicity. (2) Sentences including *part of* and indicating a subfeature. As an example, in the DigitalHome we found: *The controller part of thermostat shall provide a "set point" temperature that is used to control the flow of heat energy (by switching heating or cooling devices on or off as needed) to achieve the set point temperature.*

Another example is the following case: *The sensor part of the thermostat has a sensitivity range between* $14^{o}F$ *and* $104^{o}F$.

6 THREATS TO VALIDITY

The reported experience has had an exploratory nature, and does not claim to be a rigorous industrial case study [36]. However, it is useful to list the main threats to the validity of our results, in order to give a fair assessment of the value of the current contribution.

Construct Validity. In our evaluation, we consider numerical data about the number of variation points and ambiguity associated to requirements. However, these data are based on subjective evaluations provided in the Review phase of our data collection procedure (see Sect. 4.3). To mitigate this subjectivity threat, an Assessment phase was introduced in which a second subject reviewed the annotations produced in the Review phase. We did not compute the degree of agreement during this procedure, due to the exploratory nature of the current study.

Internal Validity. The main threat to the internal validity of the study is the involvement of the authors of this work in the Review and Assessment phase of the data collection procedure (see Sect. 4.3). We agree that the researcher bias might have played a role in the assessment. However, we argue that this threat is partially mitigated by the independent Assessment made by two authors (as the third step of the data collection procedure), and by the evidence given through the examples presented in this paper. Furthermore, other researchers can replicate our approach using the publicly available[2] QuARS tool, and using the documents employed in our evaluation (see links in Sect. 4.2).

External Validity. Our results are limited to three requirements documents. However, we argue that the documents are representative of different domains, and have different degrees of quality – e.g., the reader should notice the low number of *vagueness* defects for the ERTMS document in Table 6, which is edited by railway domain experts, while the other documents are edited by students. Furthermore, we have observed that several variability-related terms are common among the documents. Therefore, we argue that, notwithstanding the construct validity and internal validity threats, our study has the potential to be generalised to other domains, and other requirements documents.

7 CONCLUSION

In this paper, we presented an approach for variability detection in NL requirements that is based on automatically identifying ambiguities. The approach is evaluated on three requirements documents belonging to three different domains. Our results highlight that some typically vague terms (e.g., sufficient, general) and a relevant number of ambiguous constructions (e.g., those using weak verbs, and those using coordinating conjunctions) are actually indicators of variation points. This offers hints to tailor automatic ambiguity detection tools, such as QuARS, to variability detection.

One weakness of our approach is the need to involve an expert to judge the possible variability inside a defective requirement. However, when using NLP techniques to find defects in requirements, expert judgement is already needed to identify and eliminate the false positives that can be returned by the tool [35]. We claim that a company can take advantage of this work and let the expert identify also those cases in which ambiguity in requirements is due to the need to postpone choices for later decisions in the implementation of the system, and can therefore be dealt with as possible variation points. The role of the analyst is therefore not limited to the validation of requirements, but also to the elicitation of variability, in view of better market opportunities.

[2]Actually, the tool is provided upon request to the 3rd author: this allows us to keep track of the users of the tool, and to receive feedback on its usage.

REFERENCES

[1] Mathieu Acher, Anthony Cleve, Gilles Perrouin, Patrick Heymans, Charles Vanbeneden, Philippe Collet, and Philippe Lahire. 2012. On extracting feature models from product descriptions. In *Proc. of VaMoS '12*. 45–54.

[2] Vander Alves, Christa Schwanninger, Luciano Barbosa, Awais Rashid, Peter Sawyer, Paul Rayson, Christoph Pohl, and Andreas Rummler. 2008. An Exploratory Study of Information Retrieval Techniques in Domain Analysis. In *Proc. of SPLC '08*. 67–76.

[3] Chetan Arora, Mehrdad Sabetzadeh, Lionel Briand, and Frank Zimmer. 2015. Automated checking of conformance to requirements templates using natural language processing. *IEEE transactions on Software Engineering* 41, 10 (2015), 944–968.

[4] Noor Hasrina Bakar, Zarinah M Kasirun, and Norsaremah Salleh. 2015. Feature extraction approaches from natural language requirements for reuse in software product lines: A systematic literature review. *Journal of Systems and Software* 106 (2015), 132–149.

[5] Daniel M Berry, Antonio Bucchiarone, Stefania Gnesi, Giuseppe Lami, and Gianluca Trentanni. 2006. A new quality model for natural language requirements specifications. In *12th International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ)*. 115–128.

[6] Daniel M Berry and Erik Kamsties. 2004. Ambiguity in requirements specification. In *Perspectives on software requirements*. Springer, 7–44.

[7] Francis Chantree, Bashar Nuseibeh, Anne De Roeck, and Alistair Willis. 2006. Identifying nocuous ambiguities in natural language requirements. In *Requirements Engineering, 14th IEEE International Conference*. IEEE, 59–68.

[8] Kun Chen, Wei Zhang, Haiyan Zhao, and Hong Mei. 2005. An approach to constructing feature models based on requirements clustering. In *Proc. of RE'05*. 31 – 40.

[9] Lianping Chen, Muhammad Ali Babar, and Nour Ali. 2009. Variability management in software product lines: a systematic review. In *Proceedings of the 13th International Software Product Line Conference*. Carnegie Mellon University, 81–90.

[10] Horatiu Dumitru, Marek Gibiec, Negar Hariri, Jane Cleland-Huang, Bamshad Mobasher, Carlos Castro-Herrera, and Mehdi Mirakhorli. 2011. On-demand feature recommendations derived from mining public product descriptions. In *Proc. of ICSE'11*. 181–190.

[11] Fabrizio Fabbrini, Mario Fusani, Stefania Gnesi, and Giuseppe Lami. 2001. An automatic quality evaluation for natural language requirements. In *Proceedings of the Seventh International Workshop on Requirements Engineering: Foundation for Software Quality REFSQ*, Vol. 1. 4–5.

[12] Alessandro Fantechi, Stefania Gnesi, and Laura Semini. 2017. Ambiguity Defects As Variation Points in Requirements. In *Proceedings of the Eleventh International Workshop on Variability Modelling of Software-intensive Systems (VAMOS '17)*. ACM, New York, NY, USA, 13–19. https://doi.org/10.1145/3023956.3023964

[13] Henning Femmer, Daniel Méndez Fernández, Stefan Wagner, and Sebastian Eder. 2017. Rapid quality assurance with requirements smells. *Journal of Systems and Software* 123 (2017), 190–213.

[14] Alessio Ferrari, Felice Dell'Orletta, Andrea Esuli, Vincenzo Gervasi, and Stefania Gnesi. 2017. Natural Language Requirements Processing: a 4D Vision. *IEEE Software (to appear)* (2017).

[15] Alessio Ferrari, Beatrice Donati, and Stefania Gnesi. 2017. Detecting Domain-specific Ambiguities: an NLP Approach based on Wikipedia Crawling and Word Embeddings. In *2017 IEEE 25th International Requirements Engineering Conference Workshops (REW)*. IEEE, 393–399.

[16] Alessio Ferrari, Giorgio O Spagnolo, and Felice Dell'Orletta. 2013. Mining commonalities and variabilities from natural language documents. In *Proceedings of the 17th International Software Product Line Conference*. ACM, 116–120.

[17] Alessio Ferrari, Giorgio Oronzo Spagnolo, and Stefania Gnesi. 2017. PURE: A Dataset of Public Requirements Documents. In *Requirements Engineering Conference (RE), 2017 IEEE 25th International*. IEEE, 502–505.

[18] Alessio Ferrari, Paola Spoletini, and Stefania Gnesi. 2015. Ambiguity as a resource to disclose tacit knowledge. In *Requirements Engineering Conference (RE), 2015 IEEE 23rd International*. IEEE, 26–35.

[19] William Frakes, Ruben Prieto-Diaz, and Christopher Fox. 1998. DARE: Domain analysis and reuse environment. *Ann. Softw. Eng.* 5 (Jan. 1998), 125–141.

[20] Vincenzo Gervasi, Ricardo Gacitua, Mark Rouncefield, Peter Sawyer, Leonid Kof, L Ma, P Piwek, A De Roeck, Alistair Willis, H Yang, et al. 2013. Unpacking tacit knowledge for requirements engineering. In *Managing requirements knowledge*. Springer, 23–47.

[21] Benedikt Gleich, Oliver Creighton, and Leonid Kof. 2010. Ambiguity detection: Towards a tool explaining ambiguity sources. *Requirements Engineering: Foundation for Software Quality* (2010), 218–232.

[22] Stefania Gnesi, Giuseppe Lami, and Gianluca Trentanni. 2005. An automatic tool for the analysis of natural language requirements. *Comput. Syst. Sci. Eng.* 20, 1 (2005).
[23] Gregory Goth. 2016. Deep or shallow, NLP is breaking out. *Commun. ACM* 59, 3 (2016), 13–16.
[24] A Handbook. 2003. From Contract Drafting to Software Specification: Linguistic Sources of Ambiguity. (2003).
[25] Thomas B Hilburn and Massood Towhidnejad. 2007. A case for software engineering. In *Software Engineering Education & Training, 2007. CSEET'07. 20th Conference on*. IEEE, 107–114.
[26] Nili Itzik, Iris Reinhartz-Berger, and Yair Wand. 2016. Variability analysis of requirements: Considering behavioral differences and reflecting stakeholdersâĂŹ perspectives. *IEEE Transactions on Software Engineering* 42, 7 (2016), 687–706.
[27] Kyo C Kang, Sholom G Cohen, James A Hess, William E Novak, and A Spencer Peterson. 1990. *Feature-oriented domain analysis (FODA) feasibility study.* Technical Report. Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst.
[28] Mohamad Kassab, Colin Neill, and Phillip Laplante. 2014. State of practice in requirements engineering: contemporary data. *Innovations in Systems and Software Engineering* 10, 4 (2014), 235–241.
[29] Yang Li, Sandro Schulze, and Gunter Saake. 2017. Reverse Engineering Variability from Natural Language Documents: A Systematic Literature Review. In *Proceedings of the 21st International Systems and Software Product Line Conference-Volume A*. ACM, 133–142.
[30] Mikyeong Moon, Keunhyuk Yeom, and Heung Seok Chae. 2005. An approach to developing domain requirements as a core asset based on commonality and variability analysis in a product line. *IEEE transactions on software engineering* 31, 7 (2005), 551–569.
[31] Sana Ben Nasr, Guillaume Bécan, Mathieu Acher, João Bosco Ferreira Filho, Nicolas Sannier, Benoit Baudry, and Jean-Marc Davril. 2017. Automated extraction of product comparison matrices from informal product descriptions. *Journal of Systems and Software* 124 (2017), 82–103.
[32] Nan Niu and Steve M. Easterbrook. 2008. Extracting and Modeling Product Line Functional Requirements. In *Proc. of RE'08*. 155–164.
[33] Nan Niu and Steve M. Easterbrook. 2008. On-Demand Cluster Analysis for Product Line Functional Requirements. In *Proc. of SPLC'08*. 87–96.
[34] Sooyong Park, Minseong Kim, and Vijayan Sugumaran. 2004. A scenario, goal and feature-oriented domain analysis approach for developing software product lines. *Industrial Management & Data Systems* 104, 4 (2004), 296–308.
[35] Benedetta Rosadini, Alessio Ferrari, Gloria Gori, Alessandro Fantechi, Stefania Gnesi, Iacopo Trotta, and Stefano Bacherini. 2017. Using NLP to Detect Requirements Defects: An Industrial Experience in the Railway Domain. In *International Working Conference on Requirements Engineering: Foundation for Software Quality*. Springer, 344–360.
[36] Per Runeson and Martin Höst. 2009. Guidelines for conducting and reporting case study research in software engineering. *Empirical software engineering* 14, 2 (2009), 131.
[37] Helmut Schmid. 2013. Probabilistic part-ofispeech tagging using decision trees. In *New methods in language processing*. 154.
[38] Sri Fatimah Tjong and Daniel M Berry. 2013. The design of SREE - a prototype potential ambiguity finder for requirements specifications and lessons learned. In *International Working Conference on Requirements Engineering: Foundation for Software Quality*. Springer, 80–95.
[39] Thomas von der Maßen and Horst Lichter. 2003. RequiLine: A requirements engineering tool for software product lines. In *International Workshop on Software Product-Family Engineering*. Springer, 168–180.
[40] Nathan Weston, Ruzanna Chitchyan, and Awais Rashid. 2009. A framework for constructing semantically composable feature models from natural language requirements. In *Proc. of SPLC '09*. 211–220.
[41] William M Wilson, Linda H Rosenberg, and Lawrence E Hyatt. 1997. Automated analysis of requirement specifications. In *Proceedings of the 19th international conference on Software engineering*. ACM, 161–171.
[42] Hui Yang, Anne De Roeck, Vincenzo Gervasi, Alistair Willis, and Bashar Nuseibeh. 2010. Extending nocuous ambiguity analysis for anaphora in natural language requirements. In *Requirements Engineering Conference (RE), 2010 18th IEEE International*. IEEE, 25–34.
[43] Hui Yang, Alistair Willis, Anne De Roeck, and Bashar Nuseibeh. 2010. Automatic detection of nocuous coordination ambiguities in natural language requirements. In *Proceedings of the IEEE/ACM international conference on Automated software engineering*. ACM, 53–62.