Ambiguity in Requirements Engineering: Towards a Unifying Framework

Vincenzo Gervasi[1,3], Alessio Ferrari[2], Didar Zowghi[3], and Paola Spoletini[4]

[1] Dipartimento di Informatica, University of Pisa, Italy
[2] Istituto di Scienza e Tecnologie dell'Informazione "Alessandro Faedo", CNR, Italy
[3] Faculty of Engineering and IT, University of Technology, Sydney, Australia
[4] SWEGD, Kennesaw State University, USA

Abstract. A long stream of research in RE has been devoted to analyzing the occurrences and consequences of *ambiguity* in requirements documents. Ambiguity often occurs in documents, most often in natural language (NL) ones, but occasionally also in formal specifications, be it because of abstraction, or of imprecise designation of which real-world entities are denotated by certain expressions. In many of those studies, ambiguity has been considered a defect to be avoided. In this paper, we investigate the nature of ambiguity, and advocate that the simplistic view of ambiguity as merely a defect in the document does not do justice to the complexity of this phenomenon. We offer a more extensive analysis, based on the multiple linguistic sources of ambiguity, and present a list of real-world cases, both in written matter and in oral interviews, that we analyze based on our framework. We hope that a better understanding of the phenomenon can help in the analysis of practical experiences and in the design of more effective methods to detect, mark and handle ambiguity.

1 Introduction

The study of properties of software requirements specifications (SRS) has been an important and recurring theme throughout the evolution of requirements engineering (RE) research. Fundamental issues concerning the *contents* of requirements, such has how to avoid or detect inconsistencies in SRS (and whether to remove or tolerate them), or how to ensure completeness of the requirements, have been a mainstay in RE. The reasons are clear: no implementation can satisfy an inconsistent SRS, and an incomplete SRS, once implemented, will not satisfy all the needs of the users of the corresponding software system. In fact, consistency and completeness have been regarded in our earlier work [1, 2] as the two main factors for requirements correctness, i.e. as two quality features that an SRS must, eventually, possess.

Another related stream of research has been concerned with properties of the *form* of requirements, rather than of their content. Properties such as understandability, † conciseness, etc. have been studied and discussed, and techniques to ensure an SRS exhibits such properties (or to identify and fix their negative dual properties) have been proposed.

† nota molto lunga per il mio eventaule coautore, sempre che non abbia a che fare con l'assenza di un autore che abbia voglia di farmi da coautore

In this paper, we focus mainly on *ambiguity*, i.e. the phenomenon by which multiple distinct meanings can be assigned to the same requirement (or, more generally, sets

of requirements), and discuss the relationships between ambiguity and certain related phenomena which are often observed in requirements. The process of transforming an intended meaning into a set of signs (the documented form of a requirement), and then back from signs to meanings, is complex enough – even in the case of formal or diagrammatic languages – that an accurate understanding of its different facets is needed for an effective management of the requirements. Given the complexity of the ambiguity phenomenon and of its relationship with requirements, this work does not have the ambition of being a comprehensive analysis of neither of the two. Certain areas, such as for example how interactions between certain requirements can generate ambiguity that were not present in the requirements in isolation, are not dealt with in the paper.

We do not provide in this work advice on how to avoid introducing ambiguity in an SRS, nor on how to remedy it when it is detected. Rather, we focus on understanding what ambiguity *is* (with particular reference to its role in requirements engineering), on how, when and by whom it is introduced in SRS, and on what the effects of its various forms are. We posit that ambiguity is not necessarily a defect, and in fact can play an important positive role both in the requirements as a document, and in the requirements elicitation process. In short, this paper is not about a solution to a problem, but rather about exploring and characterizing the nature of a phenomenon, with supporting evidence from real-world practice.

At the same time, the paper constitutes a call to fellow researchers to draw more freely on the vast amount of knowledge that scholarly work in semiotics and linguistics has produced in the past. While requirements specifications are a very peculiar type of document, with very specific uses, and often the language employed in them is highly stylized when not entirely formalized, yet the fundamental situation is still that of two parties communicating through some form of language, and most results from established discourse theories in linguistics apply equally well to requirements engineering.

The rest of this paper is organized as follows: Section 2 discusses our basic model of ambiguity in the interpretation of RE documents (e.g., SRS); this is followed by an analysis and classification of the different levels at which ambiguity can be introduced. Section 3 presents a discussion on linguistic sources of ambiguity. Section 4 then discusses the relationships between ambiguity, abstraction and absence, and presents both a theoretical framework and a classification of different forms of ambiguity. Sections 5 and 6 present a number of real-world cases of ambiguity, in written text and in oral interviews respectively, which are characterized according to our framework. This is followed by a short survey of related works (where we restrict ourselves to the RE literature) and by some conclusion.

The present paper is a substantially extended and revised version of our previous work in [3].

2 Ambiguity and interpretation in requirements engineering

As has been recognized in the literature (see Section 7), ambiguity is a complex, multi-level phenomenon. While the general concept of "having multiple meanings" is relatively easy to describe, locating the original source – or root cause – of the ambiguity

may be challenging. Moreover, ambiguity may or may not be detected by the several parties involved in requirements elicitation or analysis, and be intentional or accidental; its extent can be confined to a minor detail or encompass some major aspect of the system.

It is clear that the simple intuitive definition of "having multiple meanings" is insufficient for a deep understanding of ambiguity. We will instead use as reference frame that of the classical denotational approach, where semantics is given by a function mapping from a source domain (the text of the requirements) to a target domain (the denotation of their semantics). Which particular semantics we want to observe (e.g., input/output semantics, performances, labeled transition system, development cost estimates, etc.) is immaterial for our discussion, and we can imagine that the semantics domain will in fact be different for different purposes[1]. Of course we will not insist on all the properties that characterize domains according to the Scott-Strachey definition [4] (e.g., that domains are partial orders, or that the mapping is Scott-continuous), nor on the compositionality of the mapping function. In requirements engineering, all these three elements — source domain, target domain, mapping function — are fuzzy at best. The source domain can include, in addition to written text, spoken information, observed behavior, references to pre-existing systems or work practices, etc. The target domain (i.e. the semantics of the requirements) should in theory be such that it is possible to determine if a given specification or implementation satisfies the requirements, but in practice it is often in itself vague. And finally, the mapping is ill-defined, and often – even when a strict formal definition exists – may well be misunderstood by at least some of the stakeholders (e.g., an end-user will probably be incapable of understanding the meaning of a fragment of Z [5] from a complex requirements specification). This last point is worth stressing. For the purpose of assessing the effects of ambiguity, it is not the *intrinsic* meaning of a requirement or set thereof that is of interest (even when we have such a thing, e.g. in formal languages), but the *interpretation* placed on it by a cognitive agent or interpreter.

We will thus accept the fuzziness of all our elements, while still keeping the general framework of denotational semantics, and asking only that any semantics, in order to be considered acceptable, must allow for testing an implementation for satisfaction of the requirements. This is in keeping with Dana Scott's position [6] that

> It is not necessary for the semantics to determine an implementation, but it should provide criteria for showing that an implementation is correct.

Figure 1 shows an overview of how we define the various steps and transformations which can lead to multiple meanings. The details will be elucidated in the next section; for now it suffices to say that the meaning of a stated requirement is determined on a purely symbolic base, based on which symbols constitute the requirement, and on the

[1] In fact, a statement can be ambiguous or not depending on which particular semantics we observe. For example, a requirement asking for a given feature in ambiguous terms could be interpreted in several different ways for the purpose of implementing the feature, and thus be ambiguous; however, if the possible interpretations all have the same implementation costs, the meaning would be unique for cost-estimate purposes, and thus no ambiguity would arise in that particular denotation.

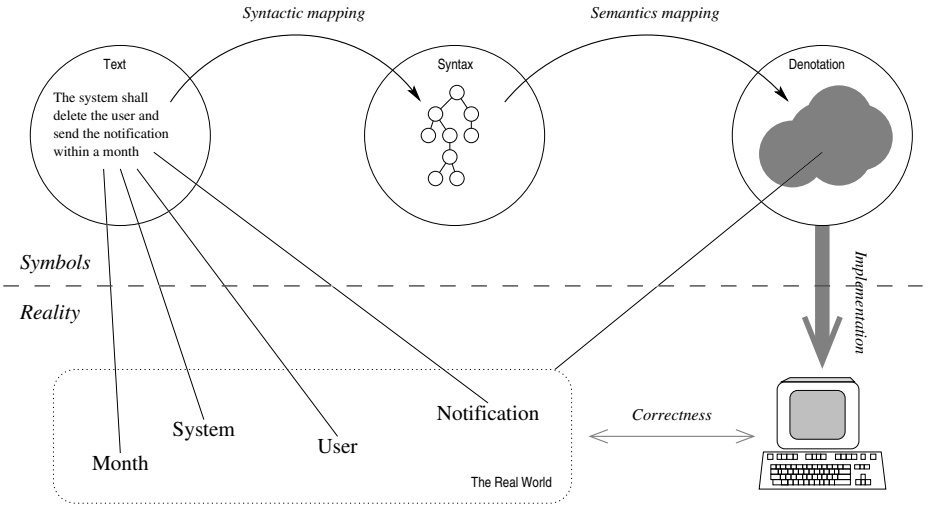

Fig. 1. The theoretical framework for the occurrence of ambiguity.

grammar and interpretation rules of the language used (in Figure 1 we are using natural language as an example, but the same holds for any other notation used to express the requirements). The denotation of the semantics of the requirements is then what drives the implementation, whose purpose is to build a computer-based system (in the real world) which will interact with its environment and modify it in such a way that the original intention expressed through the requirements is satisfied.

In the following, we outline a more structured analysis of ambiguity, with particular reference to its potential sources, and the roles that ambiguity can play in requirements specifications.

3 Linguistic sources of ambiguity

Different forms of ambiguity are introduced in a requirements document at different levels. As ambiguity is essentially a linguistic phenomenon, in that it pertains to how meaning is associated to a certain statement or set of statements in some language, it is appropriate to analyze sources of ambiguity according to the usual linguistics paradigm of lexicon, syntax, semantics. In this paper, we will not explicitly consider the fourth possible source of ambiguity, namely pragmatics, which could generate ambiguity due to contextual information (e.g. about the nature or intent of a document), or to the identities or roles of the actors in the exchange, or based on the stated or presumed intentions of the author or speaker, etc. However, these are informally addressed in the examples presented in Section 6.

The distinction is not new; in relation to requirements quality it has been introduced in [7] and maintained in subsequent studies, such as [8–10]. We will briefly outline the main issues that occur at different levels here, not delving into all the details since our main interest is only on one particular form, as will be discussed in the following.

3.1 Levels of ambiguity

Lexical level. Ambiguity in lexicon occurs typically when the same term is used to denote different things. This can be an inherent feature of the language being used (for example: homonyms in natural language, as in *bank* account vs. *bank* of a river), or happen even in more formal languages due to lack of or imprecise *designations* [11]. In fact, even in formal languages such designations are invariably rooted in the informal real world, and all stakeholders must *a-priori* agree on their meaning (thus establishing a common base of reference). Of course, this rarely happens in practice; even when using a particular domain's jargon, it is often the case that certain terms are found to be ambiguous.

It is worth remarking that even approaches based on lexical semantics (e.g., Wordnet [12]) or ontologies (e.g., LEL [13]) cannot rule out the risk of lexical ambiguity: in these approaches, relations between terms are spelled out, but the meaning of a term is only given by other terms. And after all, words are conventional and do not derive from things as Cratylus used to believe [14]. The only thing keeping Humpty Dumpty from really going by his statement "When I use a word, it means just what I choose it to mean – nothing more nor less." is the need of making himself clear to Alice [15] — unfortunately, not something to rely upon when analyzing software requirements.

In Figure 1, terms appearing in the requirement (in the Text circle), such as "user" or "month" are just lexical tokens. They can correspond to different designations, e.g. "month" could mean a 30-days period, or a 31-days period, or till the same-numbered day in the next month, or go with the moon phases, or even some fancier period[2]. Without a more precise designation, the term "month" is seriously ambiguous: for example, which date is "a month after January 30"?

Syntactic level. Ambiguity on the syntactic level is in a sense easier to define. It stems from there existing multiple parse trees for a single sentence (or, more generally, a given segment of the linear stream of language); to each possible parse tree, a different meaning is attached, hence the ambiguity. Berry, Kamsties and Krieger in [8] have provided a large number of delightful examples of ambiguities in several natural languages (and guidelines on how to avoid the most common pitfalls).

In Figure 1, multiple possible parse trees exist for our sample requirement. In fact, the sentence could be parsed as "The system shall delete the user and (send the notification within a month)" (Figure 2, left) or as "The system shall (delete the user and send the notification) within a month" (Figure 2, right), where the parentheses have been used to indicate the two critically different parsings.

Interestingly, syntactic ambiguity can be avoided by design in certain languages, and in fact most formal languages are so designed that their grammar does not allow ambiguous parsing (this allows the construction of simple and efficient parsers). This is the case in fact of almost all programming and formal specification languages, but is also claimed of some engineered spoken language, e.g. Loglan [16].

[2] The Bahá'í calendar, for example, has 19 months of 19 days each, plus 4 intercalary days (5 in leap years) which are not part of any month. How would our requirement be interpreted in a Bahá'í community?

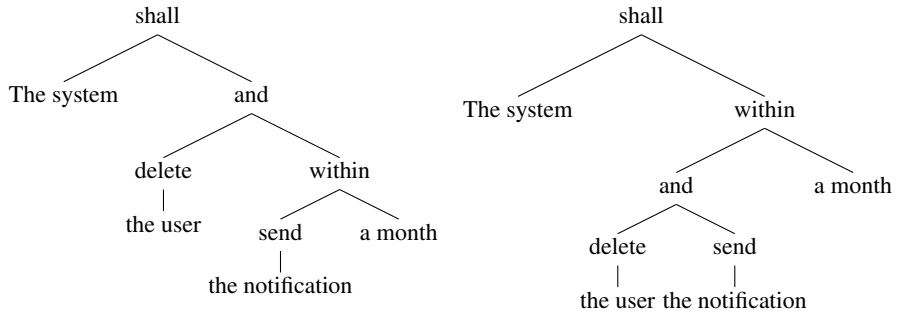

Fig. 2. Two different parse trees for our sample requirement: a case of syntactic ambiguity. This situation is termed *coordination ambiguity*.

Semantic level. Semantic ambiguity is our main concern in this paper. This happens when the source text is uniquely determined in both lexicon and syntax, i.e. the exact meaning of all terms is established, and there is only one correct parsing of the text. Nevertheless, multiple meanings can be assigned to the sentence. In this case, the ambiguity lies not in the source, but in the function assigning meaning to the source, labeled in Figure 1 as the *semantics mapping* function.

In holding this view we differ from [8, 17, 10], where semantic ambiguity is ascribed to coordination ambiguity (properly deciding the operands of *and*, *or* or other sentence constructors), referential ambiguity (resolving pronominal references and anaphora), and scope ambiguity (delimiting the scope of quantifiers). We rather consider that these phenomena stem from the syntax of the language: in fact, if natural language had some form of parentheses (for scoping) and indexing (for references), these problems would disappear (as shown, accidently, by our previous example in Figure 2 about syntactic ambiguity). In practice, typography and layout can at times serve to mark parenthetical structure when the language does not offer it: for example, we could have written our requirement as:

> The system shall:
> – delete the user, and
> – send the notification within a month.

or rather

> The system shall:
> – delete the user, and
> – send the notification
> within a month.

to make our intention clear. In fact, some systems for the automated analysis of NL requirements such as [18] have used layout to help infer parenthetical structure in such cases.

Another proof that types of ambiguity cited above are syntactic phenomena lies in the fact that semantic ambiguity can also happen in formal languages, where the lexicon and syntax are perfectly defined. In this case, the *understanding* of the semantics

of the language on the part of a reader can cause a certain statement to be interpreted ambiguously. Whether a standard semantics for the language exist or not is somewhat irrelevant to the phenomenon itself (e.g., it is often the case that even skilled programmers ignore some subtle point of the formal semantics of a programming language[3] or that the description of the semantics in itself is ambiguous). In practice, the standardized semantics of some language can be, and often is, so large and complex that it can be considered for all practical purposes to be cognitively inaccessible to the reader: hence, the semantics mapping function really used by a reader can be different from the official one, of which it will be just one of many possible approximations. And of course, different approximations by different readers, or at different times, will produce different meanings.

In Figure 1, even if we have precise designations for "month", "system", "user" etc., and even if we are told in some way which of the two syntactic interpretation to take, we could still have doubts on the intended semantics. For example, "shall send a notification" means the system will attempt to do it, but how? Is it sufficient to print out a form and hope that some operator will put it in an envelope and give to the Post Office for delivery? What if the notification is sent, but not delivered? Is there some sort of acknowledgment to be expected? Maybe the notification could be sent via a text message to the user's mobile phone? Or maybe, the notification is not intended for the user, but for his manager? And so on (endlessly).

We will devote the rest of the paper to semantic ambiguity: that is, the form of ambiguity which arises irrespective of lexicon and syntax. We will have much to say about its role in requirements elicitation and analysis.

3.2 Ambiguity vs. Vagueness

It is important not to unduly conflate *ambiguity* with *vagueness*, a different (yet related) linguistic phenomenon. Ambiguity denotes the existence of multiple distinct meanings, with an implicit assumptions that they are individually well-defined – so that solving the ambiguity means making a choice between a discrete set of possible interpretations, eventually leading to a certain implementation. In contrast, vagueness refer instead to cases where the meaning itself is fuzzily defined (Figure 3); the possible implementations form a continuous space, no longer a discrete set. Consequently, requirements cannot be said to be satisfied or not; rather, the notion of *degree of satisfaction* comes into play. This is often the case with non-functional requirements, and the topic has been extensively researched in requirements literature: in fact, both techniques to explicitly model the vagueness (e.g. by using fuzzy logic or the concept of satisficing in goal models) and recommendations to avoid vagueness (e.g., by substituting every non-functional requirement of this kind with some measurable proxy, as in service level agreements) have been advocated [19].

In natural languages, ambiguity and vagueness are often intertwined, due to one of the parties assuming a vague meaning, and the other assuming two (or more) different

[3] For a concrete example, even an experienced C language programmer might look puzzled at a statement like `long c=3["test"];` which is perfectly legal and unambiguous in the language. But then, `int x=*(char *)&c;`, which is again legal, produces results which are not specified by the semantics, and is thus ambiguous (probably, `x` would be either 0 or 116).

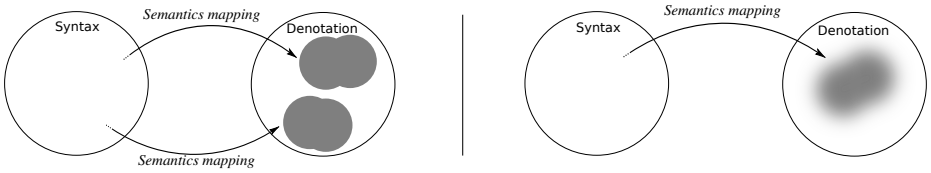

Fig. 3. A visual rendition of the difference between ambiguity (left) and vagueness (right).

points inside the continuous space to be two (or more) distinct meanings. In practice, it is not uncommon for a stakeholder to use a vague term to signify a genuine lack of preference, whereas an analyst might insist in setting on a single non-vague choice.

4 Ambiguity, Abstraction, Absence

We have seen in the previous section how even a simple sentence like our example

> The system shall delete the user and send the notification within a month.

which could appear among the old-fashioned requirements, say, for a library loan system when membership expires, is actually riddled by lexical, syntactic, and semantic ambiguity, so that its correct implementation, missing further information, is probably beyond hope.

One could then believe that ambiguity is thus a pernicious defect, to be eradicated with ruthless determination from any self-respecting requirements specification. Unfortunately, this noble determination often leads to the practical impossibility of writing down, analyzing, and implementing, the requirements for even the simplest of software systems, while huge amounts of effort are devoted to writing beautifully complex and extensive specifications.[4]

We believe instead that ambiguity can also play a positive role in requirements specifications, beyond its well-known political role in negotiations. To this end, we need first to distinguish among three related concepts:

- **Ambiguity** is the existence of multiple denotations for the same source text in the semantics space. Whether this is caused by syntactic ambiguity (as in Figure 4, bottom) or by semantics ambiguity (as in Figure 4, top), or by lexical issues (e.g., uncertain designations) is irrelevant: the essence of the phenomenon is in having multiple (distinct) semantics for the same source. Ambiguity has often been considered a defect in requirements, in account of the lack of a single, well-defined, shared semantics that can be used to drive implementation (and, later, verification).
- **Abstraction** is the omission of some details (or more properly, of some information content). Ambiguity can be used as a form of abstraction, in that the detail missing is the information needed to discriminate between multiple semantics in

[4] In fact, we rather believe that it is essentially impossible to write unambiguous specifications, with space for doubt for some purely symbolic processing system (such as lexical domains in [11]), that start with Peano axioms [20] and work up from those.

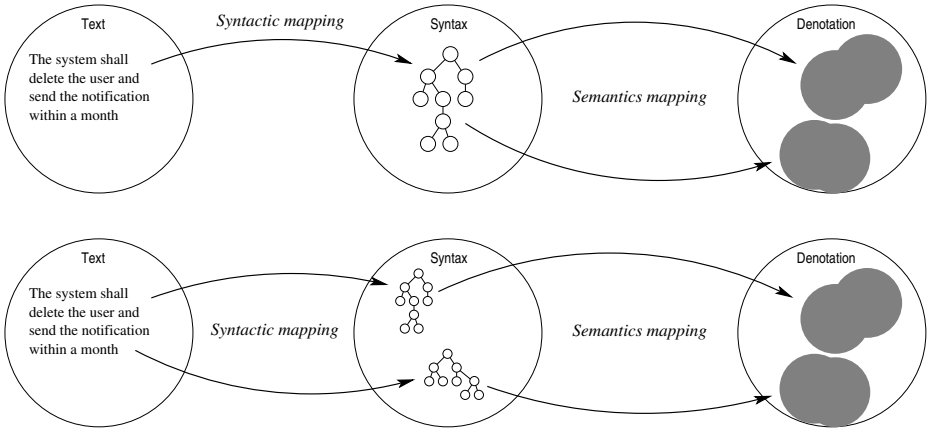

Fig. 4. Cases of ambiguity: semantics ambiguity (top), syntactic ambiguity (bottom).

order to identify the right ones (in the eye of the requirement author). Abstraction is generally considered a desirable quality in requirements, up to a point, in that it avoids overspecification, a flaw that may cause waste of time or generation of new errors [21], and thus simplifies the requirements, keeping them manageable and allowing stakeholders to focus on the important parts.

– **Absence** is the total lack of details on some specific aspect; as such, it is the extreme case of abstraction, where certain information content is abstracted to nothingness (hence, it is at times called *silence*). Being a special case of abstraction, absence as well can be related to ambiguity as discussed above. Absence is the major motivation for requirements elicitation: knowledge holes are usually considered dangerous in specifications, and need to be filled-in by investigating the problem and its domain in more depth.

In witness of the pervasiveness of ambiguity in requirements, our sample requirement also contains instances of both abstraction and absence. Using the term "month" can be seen as a not very precise way to refer to some specific duration of time, essentially conveying the idea of "I don't care about the *exact* duration, but it should be close to 30 days", hence this is a case of abstraction. At the same time, nothing is said about the actual contents of the "notification", e.g. which text should be sent. Of course, the implemented system will have to send some specific text (we cannot keep the actual message abstract in the implementation), so the missing information is needed, and hence this is really a case of absence.

Since ambiguity can play both negative and positive roles, the question arises naturally: when is bad ambiguity turned into good abstraction, and when is the latter turned again into bad absence? We believe this question, in this crude form, is too simplistic, and more about the *intentions* of the stakeholders working on and with the requirements must be considered.

As a fist step, let us identify two roles in interacting with requirements, those of *author* and of *reader*. The author is the stakeholder that commits a requirement to a

written form: it is not necessarily the customer or the problem owner, and in fact it could also be a requirement analyst, a consultant, or the long-gone author of a procedures manual that has since left the company. The reader is the participant to the development process who needs the information conveyed by the requirements in order to perform his or her own job. Implementors, testers, customer (in validation), developers of other systems, etc. can all play the role of readers.

Both writers and readers may or may not recognize the ambiguity which is present in a requirement. This gives rise to the combinations shown in Table 1.

		Reader	
		recognized	unrecognized
Writer	recognized	**(a)** ambiguity used by writer as abstraction device, recognized as such: good use of ambiguity, any implementation correct.	**(b)** writer used ambiguity as abstraction device, reader only recognized one possible meaning: loss of design space.
	unrecognized	**(c)** writer wrote ambiguous requirement without realizing, reader assumed all meanings are acceptable: potential incorrect implementation.	**(d)** ambiguity gone unnoticed: if both reader and writer agree on meaning, correct implementation possible by chance, otherwise incorrect implementation.

Table 1. Recognized and unrecognized ambiguity.

We assume for now that recognizing an ambiguity means being cognizant of all the possible meanings, whereas not recognizing it means considering only one meaning (which may or may not be the intended one), and not realizing that there is a potential ambiguity. Naturally, in practice we can be faced with fuzzy cases, in which we can suspect that there is an ambiguity but cannot determine for certain (e.g., for lexical or semantic ambiguity), or the different meanings conveyed by an ambiguous statement can themselves be vague and blend into each other without clear distinction. For the sake of exposition, we will oversimplify the issue in the following analysis.

When the ambiguity is recognized by the writer (cases (a) and (b) in Table 1), we can assume that it is *intentional*: the writer is using ambiguity as a means of abstracting away unnecessary details, signifying that all possible meanings are all equally acceptable to her as correct implementations of the requirements. For example, in our previous example from Figure 1, the clause "within a month" could be intentionally ambiguous, meaning that the writer (e.g., the customer) is not interested in the exact limit, as long as there is a fixed term, and the term is *approximately* a month. In case (a), the reader (e.g., the implementor) also recognizes the ambiguity, and is free to choose, among all possible implementations that satisfy the requirement in any of its possible ambiguous meanings, the one that best suits him: for example, a simple `limit=today()+30;` in code will suffice. In case (b), the reader may not realize that the writer has given him freedom to implement a vague notion of *month*, and might implement a full calendar, taking into account leap years and different month lengths, possibly synchronizing with time servers on the Internet to give precise-to-the-millisecond months, etc. The result-

ing implementation will be correct, but unnecessarily complex. The design space for the solution has been restricted without reason, and maybe opportunities for improving the quality of the implementation in other areas (e.g., robustness or maintainability) have been lost.

If the ambiguity is not recognized by the writer (cases (c) and (d) in Table 1), we can assume it is *not intentional*: in a sense, it has crept in against the writer's intention. Hence, only one of the possible meanings is correct, whereas others are incorrect. Of course, once the ambiguity has entered the meaning chain, it is impossible to establish which of the various possible meanings was the intended one. The implementation can still be correct, but only by chance (because, among the possible interpretations, the correct one was chosen). Moreover, when multiple readers are involved, as is the case in every real-life project, the chances of *every* reader taking up the correct interpretation by chance becomes smaller as the number of readers increases: so, this type of ambiguity will probably lead to a wrong implementation, or to a correct implementation which is tested against the wrong set of test cases, or to a correct implementation which is tested correctly but then erroneously documented in users' manuals according to a wrong interpretation, etc.

5 Ambiguity Cases in Requirements Documents

In this section we review some typical cases of ambiguity in requirements documents, based on publications available from the literature [22], and we show how such real-world ambiguities can be explained by means of the presented framework.

Pronouns: Anaphora occurs in a text whenever a pronoun (e.g., *he*, *it*, *that*, *this*, *which*, etc.) refers to a previous part of the text. The referred part of the text is normally called *antecedent*. An anaphoric ambiguity occurs if the text offers more than one antecedent options [23], either in the same sentence (e.g., *The system shall send a message to the receiver, and* ***it*** *provides an acknowledge message—it = system* or *receiver*?) or in previous sentences. The potential antecedents for the pronouns are noun phrases (NP), which can be detected by means of a shallow parser.

Whenever the ambiguity can be resolved by identifying a proper textual antecedent, we can assume this to be a type of syntactic ambiguity. However, at times a proper antecedent will be missing entirely: in such cases, *context* might provide a resolution. For example, "them" might be a signifier for "our competitors" if the document serves the role of a strategic market analysis, even if no antecedent appear in the text.

Coordinating Conjunctions: coordination ambiguity occurs when the use of coordinating conjunctions (e.g., *and* or *or*) leads to multiple potential interpretations of a sentence [24]. Two types of coordination ambiguity are considered here. The first type includes sentences in which more than one coordinating conjunction is used in the same sentence (e.g., *There is a 90 degree phase shift between sensor 1* ***and*** *sensor 2* ***and*** *sensor 3 shall have a 45 degree phase shift*). The second type includes sentences in which a coordinating conjunction is used with a modifier (e.g., *Structured approaches and platforms— Structured* can refer to *approaches* only, or also to *platforms*). Coordinating

conjunctions are invariably a case of syntactic ambiguity, as the difficulty lies with producing the correct parse tree, not with the interpretation of the meaning once the correct parse tree is provided.

Vague Terms: Vagueness is associated with the usage of terms that admit a continuous set of possible interpretations (Section 3.2), such as *minimal*, *as much as possible*, *later*, *taking into account*, *based on*, *appropriate*, etc. Typical example requirements are as follows: *In case the boolean logic evaluates the permissive state, the system shall activate a certain redundant output* – which output shall be activated?

Modal Adverbs: Modal adverbs (e.g., *positively*, *permanently*, *clearly*) are modifiers that express a quality associated to a predicate. Example of ambiguous requirements using modal adverbs are: *The system shall respond positively when the no fault is identified*—the requirement does not specify what type of message should be sent. The term "positively" is thus an abstraction device (Section 4): we state only the single property of the message we are interested in (i.e., that it will be interpreted by the receiver as the positive outcome), and not all other properties the message might have, thus leaving the implementor free in that respect.

Passive Voice: The use of passive voice is a defect of clarity in requirements, and can lead to ambiguous interpretations in those cases in which the passive verb is not followed by the subject that performs the action expressed by the verb (e.g., *The system shall be shut down*—by which actor?). Omitting the actor is a case of absence of information, and as such an opportunity for further elicitation. Also, different meanings could be intended by the writer, e.g. "The system shall be shut down *on condition*", or "The system shall be shut down *by operator*" or "It shall be possible to shut down the system *for whom*", etc.

It is interesting to observe that the rules of standard English grammar allow omitting the actor, hence this is no syntactic ambiguity. However, other languages which have an *ergative case*[5] in their grammar that cannot be omitted, would rather consider this a syntax error; or if the ergative case is unmarked, this could give rise to a proper syntactic ambiguity. The fact that different languages exhibit different cases of syntactic ambiguity should come as no surprise — and in fact, that is exactly one of the reasons in support of using controlled languages in RE.

6 Ambiguity Cases in Requirements Elicitation Interviews

In this section we present typical cases of ambiguity in requirements elicitation interviews, based on publications on the topic available from the literature [25, 26], and we show how these real-world ambiguities can be explained by means of the presented framework. The cases are presented based on typical categories of ambiguity cues in interviews, namely under-specified terms, vague terms, quantifiers and pronouns.

[5] Ergative is the grammatical case for nouns that identify the intentional agent of a verb (especially a transitive verb), often marked by a special suffix or prefix.

Under-specified terms This category includes terms with a high degree of generality, i.e., terms that identify a class of concepts or actions, but do not specify some required detail. Examples are names such as *people*, *knowledge*, *movement*, *area*, *rule*, *data*, *category*, *interface*, *thing*, *detail*, *etc.* and – less frequently – verbs such as *use*, *make*, *search*, *etc.* As such, under-specified terms are a form of abstraction, applied at the lexical level.

These terms might characterize a *specific* concept in the mind of the customer, which might not be accessible to the analyst, and that can be clarified with a more detailed specification. In other cases, they can characterize a concept that is not well defined in the mind of the customer, and that hence deserves to be made more concrete. In general, using the term "under-specified" implies a desire for a greater specificity, hence this particular designation has a negative connotation, and is often used to stigmatize cases of nocuous ambiguity. Below we present real-world examples of under-specified terms, adapted from [26].

> *Example 1. A bio-medical engineer wants to develop a system that patients can use to measure their blood pressure. The system shall include a mobile application, which sends the data about the blood pressure to the general practice doctor. When asked how blood pressure is currently measured, the customer said:* There is this device. *The analyst correctly understood that a specific device is used. The analyst thought that a precise name, or brand, for the* device *was needed, to develop an interface between the mobile phone and the device. After asking, it was clarified that the bio-medical engineer did not know the name of the device (i.e., blood pressure monitor).*

> *Example 2. A customer wants to develop a mobile application that monitors the use that she makes of her mobile phone. She said:* Maybe the system could give me also some recommendations. *The analyst thought that the term* recommendations *could have two acceptable meanings: (a) negative recommendations on applications and mobile features that she should not use; (b) positive recommendations on applications that could be downloaded, and mobile features that could be used. After clarification, the first meaning resulted correct.*

We have presented examples in which only one under-specified term is used. However, we saw also situations in which several under-specified terms are used together, possibly with vague expressions (discussed in the next sub-section), giving a too abstract level to the conversation, and causing interpretation difficulties. An interesting example is presented below.

> *Example 3. One of our customers is a public administration officer. He started the interview saying:* [I want to develop] a data-base in which there are *several* profiles of users that can access to *different* levels of information, but, most of all, can do *different* operations *depending on* their profile. *We have underlined under-specified terms, and emphasised vague ones. The analyst could not assign a clear meaning to this fragment and asked:* What is the application field? *Basically, the analyst did not have a contextual ground over which the under-specified terms could make sense. Afterwards, it was clarified that*

the application field was the monitoring and assessing of EU-funded projects. The different users were the receivers of the funds, who are required to provide evidence of their expenses, and the officer, who is required to assess the projects.

Vague Terms Vague terms are terms that admit a continuous set of possible interpretations (Section 3.2) such as *minimal*, *as much as possible*, *later*, *taking into account*, *based on*, *appropriate*, *etc.*. We already discussed these in the previous section; however, in interviews their use is generally more widespread and less damaging. In fact, vague terms are often use as an effort-saving device, so that the speaker needs not focus on retrieving the more precise term, an effort that would render the conversation less natural and impede its flow. In addition, in interviews the context is often more immediately clear to both the interviewer and the interviewee (at least, at that point in time), and moreover there is usually an underlying assumption that the material will eventually end up in written form, and at that point more precise designations may be substituted for vague terms used in the oral form.

Example 4. One of our customers wants to develop a system to automatically sketch the map of apartments. The goal is to use the system before buying an apartment, to have an idea of how the place could be rearranged. The analyst suggested a robot that follows the walls when the user visits the apartment, and provides a sketch of the map that can be visualized through a tablet. The customer asked: Can I do adjustments later? *The term* later *triggered a multiple understanding phenomenon. Indeed, the analyst could intend* later *as (a) when the user was not anymore in the apartment, e.g., to actually rearrange the map, or (b) right after the map was sketched, e.g., to account for errors made by the system. When asked, the customer specified that the first interpretation was correct.*

Notice again how, by itself, *later* is non-ambiguous: it has a single meaning (i.e., at some time subsequent the initial mapping), but the degree of vagueness was incompatible with the needs of the analyst, who (arbitrarily) chose two possible interpretations that were compatible with the vague semantics, yet more precise. We could have imagined even more compatible interpretations, e.g. (c) after the apartment is sold, maybe for tax-avoidance purposes!

Quantifiers Quantifiers are the Natural language expressions that serve to select certain elements from a typically larger set of similar elements, and are thus akin to the universal quantifier $\forall$ and the existential quantifier $\exists$ in logic. These terms include *all*, *for each*, *any*, *some*, *both*, etc.

Example 5. A customer wants to develop a virtual phone-chain*, i.e., a system that alerts her when she is more than five meters from her mobile phone. The system is composed of an application to be installed in the mobile, and by a device that the user shall wear. The customer said:* From the device I can switch off all of them*. The term* all *could be interpreted in multiple ways: the device allows to switch itself and the application simultaneously (*for all*); the*

*device can switch itself and the application off in a given sequence (*for each*); the device allows to switch itself and the application separately (*any*) based on user's choice. The first interpretation resulted to be valid.*

Pronouns Personal pronouns such as *he, she, it*, possessive pronouns as *her, his, its*, relative pronouns such as *that*, *which*, demonstrative pronouns such as *this*, *those*, *etc.*, are all potential sources of ambiguity (i.e., when the target of the reference is not uniquely determined by grammatical rules), which we have considered at the semantic level.

Example 6. A customer wants to develop an electronic business card, to be passed from the mobile of the sender to the one of the receiver by means of a Bluetooth connection. Along the discussion, the electronic business card was decided to be associated with an image, like paper business cards. He said: If we are in the same area, it gets transferred. *The analyst thought that it could be referred to the information only, or also to the image, and asked:* You want to transfer just the information, or you want also the image of the card? *The customer – quite surprisingly – replied:* Just the information.

Example 7. A real-estate appraisal expert says that, when she has to estimate the value of a property, she searches for the price of similar properties in the same area. Then, she compares the characteristics of those properties with the property under evaluation, to estimate the price. She said that her problem is that: This work takes a lot of time.[6] *The analyst assumed that the time consuming work was the comparison. But, when the analyst summarised what he understood, the customer said: No, the search [of similar properties is time consuming].*

In both these examples, the pronouns are used in an anaphoric function, i.e. they refer to a noun or noun phrase that had already been mentioned in the context. Syntactic concordance rules (e.g., the pronoun must be compatible in number and gender with the referred noun) help in resolving the reference, yet may not be sufficient to identify a single possible interpretation. More candidates can be discarded by having recourse to semantics (e.g., in our example 6 *it* can only refer to something that can be transferred via a Bluetooth connection), and if multiple possible candidates still exists, to pragmatics (e.g., also in example 6, the analyst is assuming from social context that people might be more interested in the textual details of the business card, so the options for clarification offered to the customer are just two: (a) only textual information, (b) textual information+image — but the third possible interpretation, (c) only image, is discarded on pragmatic grounds).

It is worth to notice that pronouns can also serve other functions, e.g. as deictic[7] instead of anaphora. This is particularly common in interviews. Our customer from example 6 could have said *I want this transferred.* while holding in his hand a traditional,

[6] The reader will notice that the ambiguity is not raised by the vague expression *a lot*, which appeared acceptable at that stage of the conversation.

[7] A deictic expression refers to something that only exist in the context, e.g. "here" referring to the current location of the speaker, never appearing in text.

paper-based business card, and looking down at the printed face of the card while saying it. Deictics may also introduce ambiguity, which would clearly be at the contextual level.

7 Related work

Although not one of the most popular subjects, ambiguity in requirements has received some degree of attention from researchers, especially in recent years. Not always the phenomenon has been correctly described, and at times it has been mixed up with related phenomena; also, the connection with classical studies of ambiguity in the humanities is a relatively recent acquisition.

Early studies generally have considered ambiguity in relation to completeness, i.e. only in its capacity as abstraction or absence (although often the terms used are more pertaining to vagueness), and not as an independent and significant phenomenon. Among those, Boehm [27] mentions indeterminacy as a form of incompleteness, and attributes it to missing information. There is no distinction between information that the writer might want to convey and is missing due to forgetfulness (absence), information that the writer positively wanted to omit (abstraction), and information that the writer wanted to convey, but was unable to articulate (tacit knowledge). Hence, several distinct phenomena are confused into one "indeterminacy", and the latter is itself flattened into incompleteness. In [21], Meyer lists ambiguity as one of the seven deficiencies requirements specification can suffer of. In his view, ambiguity together with inadequacies with respect to the real needs, incompletenesses, and contradictions are errors that may have disastrous effects on the subsequent development steps and on the quality of the resulting software product.

Gause and Weinberg [28] correctly identified these variations, but still defined ambiguity as related to missing information and communication errors. As causes, they cite the fact that humans make errors in observation and recall (absence), tend to leave out evident information, and generalize incorrectly (wrong abstraction); communications error that occur between writers and readers are ascribed to expression inadequacies in the writing. The fact that ambiguity can be introduced by lexicon, syntax, semantics (and that these different causes call for different remedies, given that they only consider the case of unwanted ambiguity) is not explored in [28].

A similar position is taken in the later work of Schneider et al. [29], where ambiguity is defined as

> An important term, phrase, or sentence essential to an understanding of system behavior has either been left undefined or defined in a way that can cause confusion and misunderstanding. Note, these are not merely language ambiguities such us uncertain pronoun reference, but ambiguities about the actual system and its behavior.

so ambiguity is considered either as absence (examples at the lexical and syntactic levels are provided) or as confusion. Unfortunately, the definition offered defines "ambiguity" in terms of "misunderstandings" and of "ambiguities about the system", which makes it shallow and circular, preventing a more in-depth analysis. Once again, only absence and vagueness are identified, and equated to ambiguity.

In his 2002 paper [30], Kovitz sees ambiguity as a defect, and recommends to add redundancy relating to the context (i.e., everything outside the description and its subject matter that relates to it in any way) in order to remove ambiguity. It is unclear if his view is closer to consider ambiguity as absence, and thus adding relevant material would help in that it provides more information, or if it is closer to what we have called the pragmatic source of ambiguity, in which case the added material only serves to introduce the reader to the same context. In any case, most probably the added material is not really redundant nor irrelevant, since it serves a precise purpose.

A first step in separating the different levels of ambiguity (still seen as a defect *tout court*) was taken in [7], where syntactic, structural (referring to documents' structure), semantic and pragmatic levels were identified. That stream of works then continued, eventually producing tools to identify the presence of known forms of lexical and syntactic ambiguity in NL requirements [31, 32].

Bubka et al. [33] highlighted the exaggerated attention given to ambiguity (as a defect), since ambiguous statements may be "comprehended in such a way that the intended meaning is chosen" and, hence, "it would seem that under the appropriate circumstances, there is no ambiguity."; in our framework the "appropriate circumstances" would entail a form of pragmatic ambiguity resolution.

The most complete analysis of linguistic causes of ambiguity (in RE) is probably the one in [34, 8, 17, 9], which we have already discussed. The one that more closely matches our own is that by Chantree et al. [24]: Their work, though, focuses mostly on a technique to automatically identify problematic cases of coordination ambiguity in requirements, while discounting the easy cases in which lexical statistics techniques let them judge misinterpretation unlikely. However, their distinction between *nocuous* and *innocuous* ambiguity is based only on whether misunderstandings are more or less likely, in that different interpretations are preferred by different readers, and they do not consider the intent of the writer. Similarly, their distinction between *acknowledged* and *unacknowledged* ambiguity coincides with our *recognized* and *unrecognized* ambiguity, but only on the reader's side. They do not investigate the relationship between ambiguity, abstraction, and absence, nor how ambiguity can be used purposefully for a variety of reasons (including negotiation).

A thorough analysis of tools to identify and manage ambiguity is provided in [10]; they also report on experiments that indicate that reasonable performance can be obtained in certain recognition tasks.

Other works focusing on the development of tools for ambiguity detection are those of Gleich et al. [35] and Tjong and Berry [36]. Recent advances in natural language processing technologies [37], and the rising awareness about requirements quality in industry have led to the application of this previous research in extensive industrial case studies [38, 22]. Furthermore, different companies have developed commercial tools to support automated ambiguity detection, as well as other defects or smells. Among these companies, Qualicen GmbH[8], developed Requirements Scout, a tool to analyze requirements specifications aiming to uncover requirements smells; QRA Corp[9], developed QVscribe, a tool for requirements analysis for quality and consistency; OSSENO

[8] `https://www.qualicen.de/en/`

[9] `https://qracorp.com`

Software GmbH[10], developed ReqSuite, a tool to support requirements writing and requirements analysis.

8 Conclusions

This paper presented a comprehensive exploration of the nature of the ambiguity phenomena in requirements specifications. We have conducted a thorough examination of the relationship between ambiguity and two other phenomena, that of abstraction and absence of information. Furthermore, we have explored a subtle variation of ambiguity, referred to as vagueness.

This in depth analysis of different forms of ambiguity has resulted in offering as characterization of the different levels at which ambiguity can be manifested in requirements specification documents. This systematic exploration of the ambiguity phenomena and its relationship to other relevant concepts has thus enabled us to offer a theoretical framework to study different forms of ambiguity.

We thus argue that each instance of ambiguity cannot be merely considered as useful or damaging, nocuous or innocuous, good or bad just by itself, but that these characteristics can only be defined with reference to a particular set of stakeholders — and, in particular, with reference to the original author of the requirement. We believe that our exploration and classification of ambiguity presented in this paper has achieved significant steps towards an increased understanding of the important and crucial issues in identifying and handling ambiguity is requirements specifications.

We assert that an improved understanding of the nature and effect of ambiguity can help clear the way for a more positive view of ambiguity in requirements, and suggest ways to improve the current state of practice. In particular, tools and techniques aimed at identifying instances of ambiguity in requirements could incorporate the classification presented in this paper, assisting their users focus on identifying and properly handling the different types of ambiguity, particularly critical and risky cases.

Far from being just little more than the result of unapt use of the language, ambiguity has proven in our research to be an excellent instrument to expose more subtle features, which play an important role in requirements analysis [39]. Future work will focus on better exploring and exploiting the beneficial relation between ambiguity and the elicitation of tacit knowledge [25]. Furthermore, we aim to study the relationships between intentional ambiguity and *markedness* [40, 41], a typical linguistic phenomenon that received little attention so far in RE.

Acknowledgment. The authors would like to thank Stefania Gnesi for her pioneering work on ambiguity in requirements documents, and for the many scientific collaborations with her, on several subjects, that they have enormously enjoyed along the years. The first author wishes to acknowledge the financial support of the Centre for Human-Centred Technology Design Research at UTS which partially sponsored the present work. This work was partially supported by the National Science Foundation under grant CCF-1718377.

[10] `https://www.osseno.com/en/`

References

1. Zowghi, D., Gervasi, V.: The 3Cs of requirements: Consistency, completeness, and correctness. In Salinesi, C., Regnell, B., Pohl, K., eds.: Proc. of REFSQ'02, Essener Informatik Beitrage (September 2002) 155–164
2. Zowghi, D., Gervasi, V.: On the interplay between consistency, completeness, and correctness in requirements evolution. Information and Software Technology **46**(11) (2004) 763–779
3. Gervasi, V., Zowghi, D.: On the role of ambiguity in RE. In: Proceedings of the 16th International Conference on Requirements Engineering: Foundation for Software Quality, Essen, Germany, Springer-Verlag (2010)
4. Scott, D., Strachey, C.: Toward a mathematical semantics for computer languages. Technical Report PRG-6, Oxford Programming Research Group (1971)
5. ISO: Information Technology – Z Formal Specification Notation – Syntax, Type System and Semantics. ISO (2002)
6. Scott, D.S.: Lambda calculus: Some models, some philosophy. In Barwise, J., Keisler, H.J., Kunen, K., eds.: The Kleene Symposium, North-Holland Publishing Company (1980) 223–265
7. Fabbrini, F., Fusani, M., Gervasi, V., Gnesi, S., Ruggieri, S.: On linguistic quality of natural language requirements. In Dubois, E., Opdahl, A.L., Pohl, K., eds.: Proc. of REFSQ'98, Pisa, Italy, Presses Universitaires de Namur (June 1998) 57–62
8. Berry, D., Kamsties, E., Krieger, M.: From contract drafting to system specification: Linguistic sources of ambiguity (2003)
9. Berry, D., Bucchiarone, A., Gnesi, S., Lami, G., Trentanni, G.: A new quality model for natural language requirements specifications,. In: Proc. of REFSQ'06, Luxembourg, Springer-Verlag (2006)
10. Kiyavitskaya, N., Zeni, N., Mich, L., Berry, D.M.: Requirements for tools for ambiguity identification and measurement in natural language requirements specifications. Requir. Eng. **13**(3) (2008) 207–239
11. Jackson, M.: Problem Frames. Addison Wesley, Harlow, UK (2001)
12. Fellbaum, C., ed.: WordNet: An Electronic Lexical Database. MIT Press (1998)
13. Breitman, K.K., do Prado Leite, J.C.S. Number 2940 in Lecture Notes in Computer Sciences. In: Lexicon Based Ontology Construction. Springer-Verlag (2004) 41–45
14. Plato: Cratylus. In Cooper, J.M., Hutchinson, D.S., eds.: Plato Complete Works. Hackett Publishing (1997)
15. Carroll, L.: Through the Looking-Glass, and What Alice Found There. Macmillan (1871) (pseudonym of C. L. Dodgson).
16. Rice, S.L.: Loglan 3: Understanding Loglan. Master's thesis, University of Alaska at Fairbanks (May 1994) (Reprinted in serialized form by the Loglan Institute, Inc. in *La Logli* issues 1997/1, 1997/2 and 1997/3).
17. Berry, D.M., Kamsties, E.: Ambiguity in requirements specifications. In do Prado Leite, J.C.S., Doorn, J.H., eds.: Perspectives on software requirements. Volume 753 of The Kluwer international series in engineering and computer science. Springer (2004) 7–44
18. Gervasi, V.: Environment Support for Requirements Writing and Analysis. PhD thesis, University of Pisa (March 2000)
19. Chung, L., Nixon, B., Yu, E., Mylopoulos, J.: Non-functional requirements in software engineering. Kluwer Academic Publishers, Massachusetts (2000)
20. Peano, G.: Arithmetices principia, nova methodo exposita. Fratres Bocca, Turin (1889) (in Latin).
21. Meyer, B.: On formalism in specifications. IEEE Software **2**(1) (1985) 6–26

22. Ferrari, A., Gori, G., Rosadini, B., Trotta, I., Bacherini, S., Fantechi, A., Gnesi, S.: Detecting requirements defects with NLP patterns: an industrial experience in the railway domain. Empirical Software Engineering **23**(6) (2018) 3684–3733
23. Yang, H., Deroeck, A., Gervasi, V., Willis, A., Nuseibeh, B.: Extending nocuous ambiguity analysis for anaphora in natural language requirements. In: Proceedings of the 18th IEEE International Requirements Engineering Conference, Sydney, Australia (2010)
24. Chantree, F., Nuseibeh, B., de Roeck, A., Willis, A.: Identifying nocuous ambiguities in requirements specifications. In: Proceedings of 14th IEEE International Requirements Engineering conference (RE'06), Minneapolis/St. Paul, Minnesota, U.S.A. (September 2006)
25. Ferrari, A., Spoletini, P., Gnesi, S.: Ambiguity and tacit knowledge in requirements elicitation interviews. Requirements Engineering **21**(3) (2016) 333–355
26. Ferrari, A., Spoletini, P., Gnesi, S.: Ambiguity cues in requirements elicitation interviews. In: 2016 IEEE 24th International Requirements Engineering Conference (RE), IEEE (2016) 56–65
27. Boehm, B.: Some experiences with automated aids to the design of largescale reliable software. IEEE Transactions on Software Engineering **1**(1) (1975) 125–133
28. Gause, D.C., Weinberg, G.M.: Exploring Requirements: Quality Before Design. Dorset House, New York, USA (1989)
29. Schneider, G.M., Martin, J., Tsai, W.T.: An experimental study of fault detection in user requirements documents. ACM Transactions on Software Engineering and Methodology **1**(2) (1992) 188–204
30. Kovitz, B.: Ambiguity and what to do about it. In: Proceedings of the 10^{th} International Conference on Requirements Engineering, Los Alamitos, CA, USA, IEEE Computer Science Press (2002)
31. Fabbrini, F., Fusani, M., Gnesi, S., Lami, G.: An automatic quality evaluation for natural language requirements. In: Proceedings of REFSQ'01, Interlaken, Switzerland (2001)
32. Gnesi, S., Lami, G., Trentanni, G., Fabbrini, F., Fusani, M., et al.: An automatic tool for the analysis of natural language requirements. International Journal of Computer Systems Science and Engineering **20**(1) 53–62
33. Bubka, A., Gorfein, D.S.: Resolving semantic ambiguity: An introduction. In Gorfein, D.S., ed.: Resolving Semantic Ambiguity. Springer New York (1989) 3–12
34. Kamsties, E., Berry, D., Paech, B.: Detecting ambiguities in requirements documents using inspections. In: Workshop on Inspections in Software Engineering (WISE'01), Paris, France (2001) 68–80
35. Gleich, B., Creighton, O., Kof, L.: Ambiguity detection: Towards a tool explaining ambiguity sources. In: International Working Conference on Requirements Engineering: Foundation for Software Quality, Springer (2010) 218–232
36. Tjong, S.F., Berry, D.M.: The design of sreea prototype potential ambiguity finder for requirements specifications and lessons learned. In: International Working Conference on Requirements Engineering: Foundation for Software Quality, Springer (2013) 80–95
37. Ferrari, A., DellOrletta, F., Esuli, A., Gervasi, V., Gnesi, S.: Natural language requirements processing: a 4d vision. IEEE Software **34**(6) (2017) 28–35
38. Femmer, H., Fernández, D.M., Wagner, S., Eder, S.: Rapid quality assurance with requirements smells. Journal of Systems and Software **123** (2017) 190–213
39. Ferrari, A., Spoletini, P., Gnesi, S.: Ambiguity as a resource to disclose tacit knowledge. In Zowghi, D., Gervasi, V., Amyot, D., eds.: 23rd IEEE International Requirements Engineering Conference, RE 2015, Ottawa, ON, Canada, August 24-28, 2015, IEEE Computer Society (2015) 26–35
40. Merlini Barbaresi, L.: Markedness in English Discourse: A semiotic approach. Edizioni Zara, Parma, Italy (1988)
41. Chandler, D.: Semiotics: The Basics. 2nd edn. Routledge, London, UK (2007)