Requirements Eng
DOI 10.1007/s00766-016-0249-3

RE 2015

Ambiguity and tacit knowledge in requirements elicitation interviews

Alessio Ferrari[1] · **Paola Spoletini**[2] · **Stefania Gnesi**[1]

Received: 11 November 2015 / Accepted: 14 March 2016

Abstract Interviews are the most common and effective means to perform requirements elicitation and support knowledge transfer between a customer and a requirements analyst. Ambiguity in communication is often perceived as a major obstacle for knowledge transfer, which could lead to unclear and incomplete requirements documents. In this paper, we analyze the role of ambiguity in requirements elicitation interviews, when requirements are still tacit ideas to be surfaced. To study the phenomenon, we performed a set of 34 customer–analyst interviews. This experience was used as a baseline to define a framework to categorize ambiguity. The framework presents the notion of ambiguity as a class of four main sub-phenomena, namely unclarity, multiple understanding, incorrect disambiguation and correct disambiguation. We present examples of ambiguities from our interviews to illustrate the different categories, and we highlight the pragmatic components that determine the occurrence of ambiguity. Along the study, we discovered a peculiar relation between ambiguity and tacit knowledge in interviews. Tacit knowledge is the knowledge that a customer has but does not pass to the analyst for any reason. From our experience, we have discovered that, rather than an obstacle, the occurrence of an ambiguity is often a *resource* for discovering tacit knowledge. Again, examples are presented from our interviews to support this vision.

Keywords Requirements engineering · Requirements elicitation · Interviews · Ambiguity · Natural language

1 Introduction

Requirements elicitation is the process of discovering requirements for a system by accessing available knowledge sources and by communicating with the stakeholders who have a direct or indirect influence on the requirements [18, 74]. Among the available requirements elicitation techniques (e.g., workshops, focus groups, scenarios, prototypes [64, 81]), interviews with stakeholders are the most commonly used [1, 11, 24, 36] and are considered among the most effective for knowledge transfer [16, 17, 39, 75]. Normally, requirements elicitation interviews involve two roles: a customer and a requirements analyst. Several factors were observed to negatively affect the interview process, from the trustworthiness and motivation of the customer, to the absorptive capacity of the requirements analyst [18]. Among these factors, ambiguity in communication is regarded as a major obstacle [18] for knowledge transfer, since incorrectly understood needs or domain aspects might lead to the definition of poor requirements, which can cause problems in later stages of development [2].

Past works on ambiguity in requirements engineering are mainly focused on natural language (NL) ambiguities in requirements documents (i.e., textual documents) [3, 9, 10, 12, 13, 21, 22, 30, 33, 44, 45, 49, 50, 52, 54, 77–79]. Part of these works is focused on the identification of typical ambiguous terms and constructions [9, 10, 30, 33, 77]. Other

✉ Alessio Ferrari
alessiofer@gmail.com; alessio.ferrari@isti.cnr.it

Paola Spoletini
pspoleti@kennesaw.edu

Stefania Gnesi
stefania.gnesi@isti.cnr.it

[1] CNR-ISTI, Pisa, Italy

[2] Kennesaw State University, Kennesaw, GA, USA

Published online: 26 March 2016

works address the ambiguities by translating the requirements into formal languages or models [3, 13, 45]. Finally, some works focus on the usage of NL understanding methodologies [44, 52] and on artificial intelligence techniques [21, 22, 78, 79]. However, all these works study ambiguity at the level of written NL requirements, and the role of ambiguity in elicitation interviews that use NL in its oral form has not been thoroughly investigated yet.

The work presented in this paper aims at filling this gap, with the rationale that understanding ambiguity in interviews, which precede the definition of requirements documents, can cast new light on the concept of ambiguity in textual requirements. To this end, we decided to directly observe the occurrence of ambiguity by simulating a set of realistic interviews between a requirements analyst and a set of customers who wish to develop novel software-intensive products. From this study, we saw that the concept of ambiguity in NL requirements documents, and its classical lexical, syntactic, semantic clues [10], were accounting for a very limited set of ambiguity phenomena that occur at the level of requirements elicitation, in which the *pragmatic*, contextual aspect appeared to be dominant. Therefore, we defined a framework to categorize ambiguities in requirements elicitation interviews, on the basis of the work performed by Gervasi et al. [28] on *tacit knowledge*. Tacit knowledge in requirements engineering [18, 28, 75] is defined as the knowledge that a customer has but does not pass to the requirements analyst for any reason. Tacit knowledge is regarded as a major problem in requirements elicitation, and though process solutions exist [75], means are required to *improve* the detection of tacit knowledge. In our study, we found that the phenomenon of ambiguity, correctly perceived as a dangerous issue in requirements documents, is actually a powerful tool to discover tacit knowledge during requirements elicitation. Indeed, when the analyst explicitly reveals the presence of an ambiguity during an interview, the ambiguity often works as a conversational picklock to lead to the disclosure of tacit knowledge. This finding can be employed by requirements engineers to define practices that *leverage* ambiguities in requirements elicitation, and use this communication defect to achieve an improved shared understanding of the problem domain [32].

This work is an extension of a previous conference paper [23]. With respect to the original paper, the current work adds the following relevant contributions: (a) an extension of the ambiguity framework, based on the analysis of additional interviews performed with domain experts. These interviews were performed to further investigate the role of domain knowledge in the perception of ambiguity; (b) an explicit integration of the notion of innocuous ambiguity [12] within the framework; (c) a larger set of examples; (d) a quantitative view on the different types of ambiguity that we identified, and on the different types of tacit knowledge that we disclosed; (e) a throughout literature review.

The paper is organized as follows. Section 2 informally defines ambiguity in requirements elicitation interviews and presents the contextual aspects that are useful to understand our vision of ambiguity. In Sect. 3, we more formally define ambiguity by instantiating the framework for tacit knowledge defined by Gervasi et al. [28] in the context of customer–analyst interviews. Section 4 describes the different categories of ambiguities in interviews. Section 5 explains the role of ambiguity in disclosing tacit knowledge. Section 6 presents the research challenges that this work opens. Section 7 discusses related works, and Sect. 8 concludes the paper.

2 Context

This paper aims to give an insight on ambiguity in requirements elicitation interviews. In this section, we give an informal definition of ambiguity in interviews, briefly describe the performed interviews and present the fundamental concepts useful to understand the phenomenological framework that we defined for ambiguity.

2.1 Ambiguity in interviews

Requirements elicitation interviews normally involve a customer and a requirements analyst, and the elicitation process consists of a dialog in which the customer expresses his/her needs, while the analyst asks questions to identify the requirements for the system as well as domain-related aspects. Interviews are normally classified into three types, namely structured, unstructured and semi-structured [81]. In this work, we focus on unstructured interviews, in which the customer is free to talk and is not guided by a predefined set of questions.

In general, a NL expression is ambiguous when it can be interpreted in different ways. In interviews, ambiguities are associated with *misunderstanding* situations, when an expression of a customer is either not understood or incorrectly interpreted by the requirements analyst. This latter phenomenon is normally referred as subconscious disambiguation [27]. Let us give an informal definition of ambiguity in requirements elicitation interviews (a more formal definition is given in Sect. 3.4).

Ambiguity An ambiguity occurs in a requirements elicitation interview when a customer articulates a unit of information, and the meaning assigned by the requirements analyst to the articulation differs from the meaning intended by the customer.

With the term *unit of information*, we refer to two types of information that the customer might wish to articulate along the interview: system needs and domain-related aspects. Moreover, by *articulation of a unit of information*, we mean the speech fragment that expresses a system need or a domain aspect. In this sense, an articulation is a reification of a unit of information. Moreover, a speech fragment is intended here as any spoken consecutive set of words. Our definition of ambiguity includes also those cases in which the analyst *cannot* assign any meaning to the speech fragment expressed by the customer.

Notice that our definition of ambiguity takes into account only the ambiguity cases associated with expressions of the customer that are misunderstood by the analyst, and does not consider the situations in which the customer does not understand questions or comments of the requirements analyst. This choice is driven by the idea that relevant information about the system-to-be (i.e., requirements and domain knowledge) comes from the customer. Though this is not always true, since the requirements analyst can contribute to the requirements elicitation process through direct negotiation and thanks to his/her previous domain knowledge [36], this simplification avoids us to consider aspects related to dialog and argumentation—addressed by Corvera et al. [15]—which would be beyond our scope.

2.2 Interviews

The definition of ambiguity given above was used as a reference to perform our inquiry concerning ambiguity in requirements elicitation interviews. To study the problem, we simulated 34 unstructured interviews, in which the customers were asked to come to the meeting with one or more ideas of software-intensive systems to develop. The interviews were performed by the same requirements analyst (i.e., the first author), whose previous research focused on the detection of ambiguity in textual requirements [21, 22], to have a uniform perception of the ambiguity cases. The role of the customer was played by 11 domain experts—in History of Arts, Public Administration, Healthcare,[1] Training Courses, Mechanical Engineering, Statistics, Agronomy, Real-estate Appraisal and Literature—and 7 software engineers. The domain experts were asked to provide ideas for novel software-intensive systems in their domains, while the software engineers could bring novel ideas in domains in which they felt familiar. We decided to have also software engineers in the interviews to enact peer-based situations, in which both the customer and the analyst have a computer science background. These situations are rather common in some companies, in which there is an IT specialist who acts as mediator between the company and the supplier of a software product. Given our focus on the linguistic aspect of ambiguity, no graphical language was allowed during the interviews, while body language was in general unavoidable. At the beginning of each interview, the requirements analyst asked the customer to speak about his/her ideas and, when domain-related aspects emerged that appeared new to the analyst, the analyst asked for further insights. During each interview, the analyst took textual notes concerning the requirements and annotated situations that he perceived as ambiguous. Moreover, interviews were tape-recorded.

From the interviews, we isolated the speech fragments that were perceived as ambiguous by the analyst (232 in total) by extracting them from the interview notes and the tape recordings. Accurate inspection of these fragments, self-reflection and joint discussions allowed us to come to the definition of our categorization of ambiguities in interviews.

The interviews were not meant to be a rigorous empirical study, but were used as a baseline to first identify and then categorize ambiguities. Therefore, the current paper shall be regarded as a *vision* paper, in which the authors express their understanding of the ambiguity phenomenon, as it emerged from a set of observations made in a set of arranged customer–analyst interviews. In this sense, the quantitative data reported in Sects. 4.5 and 5.3 shall be read as merely informative for the reader, and not as an empirical validation of our vision. On the other hand, we strongly believe that it would have hardly been possible to develop the point of view that we are sharing in this paper without putting ourselves in the scene. Overall, this is a first step toward a broader research, as envisioned in the challenges described in Sect. 6. However, we argue that the description given of the settings of our interviews can allow other researchers to wear the lenses of our vision or to refute it, taking into account the story that generated it.

2.3 The pragmatic facet

In our on-field observation during the interviews, a first intuitive finding was that ambiguities in requirements elicitation appeared as different from the ambiguities discussed in the literature of NL requirements. As in textual requirements, some ambiguities were triggered by vague terms (e.g., “as possible” [10, 33]), some were due to the usage of universal quantifiers (e.g., “all” and plurals [9]), and few were anaphoric ambiguities [79]. However, most of the ambiguities experienced were related to the *context* of the interview, and in particular to the mental context of the requirements analyst.

[1] Three professionals in different subfields, namely Bio-medical Devices, Heath-care Management and General Medicine.

Interpretation, acceptance and access To understand these situations, it is useful to refer to Fig. 1, in which we give a model of the understanding of a speech fragment by a requirements analyst. In a perfect communication sequence, when a customer articulates a unit of information, the analyst listens to the speech fragment of the customer and accesses the expressed unit of information. Accessing the information means that the expression of the customer is well understood, and no ambiguity—defined as in Sect. 2.1—is perceived by the analyst. The access to the information (Access line in Fig. 1) implies that the analyst first gives an interpretation (Interpretation block) and then considers if this interpretation is acceptable in light of his/her current mental framework (Acceptance block). Indeed, as highlighted by Pitts and Brown [59], and as we experienced in our interviews, during the elicitation the analyst builds a mental framework of the problem domain, which is incrementally updated while new information comes from the customer. This mental framework includes the other requirements currently expressed by the customer (Requirements), the motivations of the requirements (Goals, in goal-oriented requirements engineering terms [47]), the domain knowledge currently available (Domain) and some form of mental specification of the system (Specification), which the analyst defines to assess the feasibility of the system in advance.

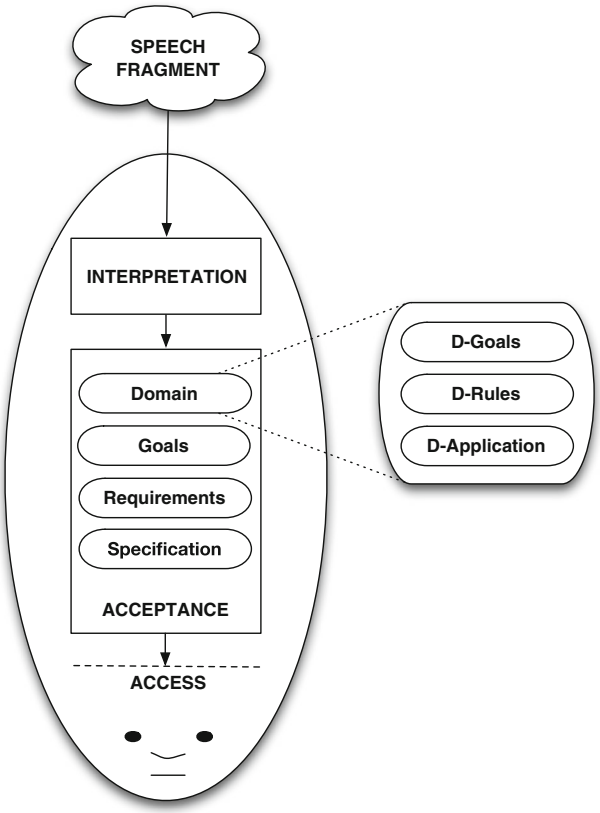

Fig. 1 A model of the process of access to a unit of information by a requirements analyst

Domain knowledge component The domain knowledge component can be further partitioned into sub-components. Indeed, when speaking with the customer, the requirements analyst tries to get information about the domain and tends to create a mental knowledge base according to three main facets. The first facet is composed by the goals inherent to the domain (D-Goals). These goals do not necessarily overlap with the goals that the system to be developed is required to satisfy. Indeed, the domain might have multiple problems to address, which could be discussed along the interview, and the system can solve only part of them.[2] The second facet is composed by the rules that regulate the domain, as understood by the analyst (D-Rules). Here, rules are intended as *domain rules*, both in terms of regulations (i.e., norms to follow within the domain) and in terms of business rules, i.e., any statement of the customer that characterize or constrain the domain, both in terms of structure and in terms of behavior [38]. The third facet is the application view that the analyst mentally builds to visualize and make sense of the information about the domain received by the customer (D-Application). This application view involves both real-world objects and practical operations. In a sense, this three-dimensional knowledge base helps the analyst in describing the domain as the *system-as-is*, a world made of goals, rules, real objects and practical operations, in which the *system-to-be* needs to be integrated.

These components *jointly* operate in the mental framework of the analyst to accept or reject the given interpretation. Indeed, to accept the unit of information expressed by the customer, the analyst compares his/her interpretation of the speech fragment with these components, to check whether the new information is consistent with his/her current understanding of the problem domain. For example, if the interpreted unit of information contradicts the requirements previously expressed, the analyst will perceive an ambiguity and will ask further clarification to the customer. In other cases, the analyst can give an interpretation to the speech fragment, but this interpretation might be different from the intended meaning of the customer. If this interpretation is acceptable in the mental framework of the analyst, he/she will not detect the ambiguity and will access to a unit of information that was not actually formulated by the customer.

[2] For example, one of the goals of the General Medicine domain is to provide treatments for the patients. A system whose goal is to support a physician in the diagnosis of a disease (as, e.g., in Example 3.3) only *contributes* to the domain goal of treating the patients. Satisfying this domain goal requires other sub-goals to be addressed (e.g., selecting medications), which are outside the scope of the system.

To account for the different types of situations that we saw in practice, which depend on the interaction of the different blocks depicted in Fig. 1, the concept of *ambiguity* needs to be defined in a precise way, and further refined. To this end, we instantiate the framework for tacit knowledge defined by Gervasi et al. [28], in the context of customer–analyst interviews, and we extend it to account for the different cases of ambiguities that we encountered in practice. In the following, we will give a definition of ambiguity, and we will provide a categorization of the phenomenon.

3 Definition of ambiguity

The framework defined by Gervasi et al. [28] aims at providing a phenomenology of tacit knowledge. To this end, the framework defines a model of the communication between stakeholders involved in a software project. The model is based on a set of predicates that can be associated with a unit of information k. This unit of information is regarded as *any desire, intention, judgement, belief, fact, reasoning rule or algorithm, which is held by a person or conveyed by a document* [28]. In requirements interviews that involve a customer and a requirements analyst as stakeholders, k represents any information concerning the system needs or the domain knowledge associated with the system to develop. At this stage, we are not interested in the notion of tacit knowledge, which is discussed in Sect. 5. Instead, we are interested in the predicates employed to represent the communication process between customer and analyst. These predicates are useful to formally reason on the dialog between customer and analyst.

Let k be a unit of information, c be the customer, a denote the requirements analyst and i be the articulation of k expressed by the customer.[3] Consider that k can belong to the requirements for the system or to the domain of the system. Moreover, consider i as a speech fragment.

From the predicates associated with k, we select two predicates that are useful for our definition of ambiguity:

- $articulated_{c,i}(k)$: a unit of information k was expressed in a speech fragment i by the customer c;
- $accessible_{a,i}(k)$: the unit of information k expressed through i was correctly accessed by the analyst a.

As in the work of Gervasi et al. [28], $accessible_{a,i}(k)$ implies that k is accessible in reasoning, or acting, or in some form of decision making. In the following subsections, we refine this latter predicate to give a first insight on the ambiguity phenomenon.

[3] Gervasi et al. [28] have i refer to the whole *interview*. Here, i is associated with the specific piece of the interview (i.e., the speech fragment) in which k is articulated.

3.1 Accessible

To refine the $accessible_{a,i}(k)$ predicate, here, it is useful to specify that k is accessible to a if the expression of k is both *interpretable* (i.e., a can assign a meaning to i) and *acceptable* in the current mental framework of the analyst. More formally:

$$accessible_{a,i}(k) = interpretable_{a,i}(k) \wedge acceptable_{a,i}(k)$$

The $interpretable_{a,i}(k)$ and $acceptable_{a,i}(k)$ predicates are novel predicates introduced by the current work. Note that $acceptable_{a,i}(k) \Rightarrow interpretable_{a,i}(k)$, since the information can be accepted only after it has been interpreted.

3.2 Interpretable

The expression of k is interpretable by the analyst (i.e., $interpretable_{a,i}(k)$) if the analyst can give at least one interpretation (correct or incorrect) to the terms, to the syntax and to the semantics of the speech fragment. From our experience, there are several cases that might cause $\neg interpretable_{a,i}(k)$. Here, we give three examples.

Example 3.1 ($\neg interpretable_{a,i}(k)$) The customer might use domain-specific terms that the analyst does not know. Consider the case of one of our customers, who is an expert in History of Arts, and wishes to realize a system for associating the paintings to the authors, to support the process of attribution. He says that, to perform attribution of a painting to an artist, he applies the *connoisseurship method* (a method based on using previous paintings to analyze the style and the themes of an artist). This domain-specific term was unknown to the analyst, and he had to ask further insight on this topic to understand the process associated with the *connoisseurship method*.

Example 3.2 ($\neg interpretable_{a,i}(k)$) The analyst might not understand the expression of the customer, because the latter is using vague terms, without a precise semantics. Consider the case of one of our customers who wishes to develop a mobile application that uses augmented reality to paint the walls of a room. During the interview, the customer says: *[The app changes the color of the wall] taking light into account*. The analyst could not understand the vague expression *taking light into account* and asked for further clarifications. Then, he understood that the customer wanted to preserve the shades of the walls due to different lightning situations.

Example 3.3 ($\neg interpretable_{a,i}(k)$) The analyst might not understand the expression of the customer, because the latter is expressing too much information at once in a chaotic way, possibly using domain-specific terms. The

analyst has too much information to process and he/she is not able to assign a meaning to the expression of the customer. For example, one of our customers is a physician who wants to develop a system to predict the dermal disease of a patient based on the observation of symptoms. He said that he wanted *Software that identifies the various diseases, and hence also the cutaneous symptoms, also for diseases that are systemic*. No meaning could be assigned to this fragment by the analyst. First, it was clarified the domain-specific meaning of *systemic disease* (i.e., a disease that affects the body as a whole). Then, it was clarified that the customer wanted a system that takes as input a series of symptoms and produces the most likely disease as output. Symptoms could be specific cutaneous manifestations, and other types of symptoms, e.g., fever and high heartbeat. The predicted disease could be dermal or systemic.

3.3 Acceptable

The fact that the analyst can give an interpretation to the speech fragment of the customer does not imply that the fragment is acceptable in the analyst's mental framework (i.e., $acceptable_{a,i}(k)$). As previously specified, the mental framework of the analyst is composed of multiple components that are involved in the acceptance of the speech fragment of the customer. To account for these components, we define:

$$acceptable_{a,i}(k) = acceptable^{G}_{a,i}(k) \wedge acceptable^{R}_{a,i}(k) \wedge acceptable^{D}_{a,i}(k) \wedge acceptable^{S}_{a,i}(k)$$

Here, G are the motivations currently associated with the system (i.e., its goals), R are the requirements currently expressed in the interview, D is the domain knowledge available to the analyst and S is the mental specification of the system. Let us exemplify some cases in which we have $\neg acceptable_{a,i}(k)$.

Example 3.4 ($\neg acceptable^{G}_{a,i}(k)$) These are the cases in which the analyst cannot understand the goal or rationale of the requirement currently expressed by the customer. For example, one of our customers wanted a system to know the time until a bus arrival. He specified that he wanted *a kind of mapping between the time left and where the bus is*. The analyst could not understand the *goal* of knowing the exact position of the bus, since, in his current mental framework, the only goal was to know how much time was left for the next bus. The customer explained that he wanted to know whether the waiting time was due to the distance of the bus from the bus stop or to traffic congestion. The hidden goal of the system was to let the user choose another means of transport, possibly passing from another street, in case of traffic congestion.

Example 3.5 ($\neg acceptable^{R}_{a,i}(k)$) The typical cases that make a requirement not acceptable with respect to the previously expressed requirements are the situations of contradiction and inconsistency. For example, one of our customers wanted to have an intelligent windshield wiper that worked according to *tapping* commands of the driver. He first said: *It would be nice to have a voice control or tap control*. The tap control was understood by the analyst as a manual tapping (i.e., the driver taps with his/her hand, and the windshield wiper moves). But then, the customer said: *I do not want to use the hands*. This was perceived as a contradiction with the previously expressed need. After asking for clarifications, the analyst understood that, with the expression *tap control*, the customer intended "tapping with the voice": He wanted to control the system by producing a sound with the voice similar to the sound that would be produced by tapping with the fingers. Basically a previous ambiguity (an *incorrect disambiguation phenomenon*, see Sect. 4.3) was discovered thanks to another ambiguity (i.e., an *acceptance unclarity*, see Sect. 4.1).

Example 3.6 ($\neg acceptable^{D}_{a,i}(k)$) These are situations in which the speech fragment of the customer is inconsistent with the domain knowledge of the analyst. In our interviews, one of the customers wanted a recycling-support system that, given the envelope of a product, tells the user in which trash bin should be thrown. The customer said: *If you do not recycle a certain thing because the municipality did not signal that it was recyclable, you will not get a fine*. From the domain experience of the analyst in recycling, incorrect recycling was not punished with any fine. Therefore, the requirements was inconsistent with his domain knowledge, and he asked clarifications. After some discussion, he understood that, in the municipality of the customer, trash bins are placed within the *condominia*, and the residents get fines from the municipality if they do not recycle properly.[4]

Example 3.7 ($\neg acceptable^{S}_{a,i}(k)$) These situations occur when there is inconsistency between a statement of the customer and the specification that the analyst mentally builds during the interview. For example, one of our customers wanted to have a swim-keeper device, to monitor his swimming training. He said: *It would be nice to show also how many strokes you take in one lap*. The analyst thought that the length of a lap could vary and that, from the specification point of view, an approach for indicating

[4] The speech fragment in this example could be seen as inconsistent with the commonsense knowledge of the analyst. However, deciding whether something is commonsense knowledge or domain knowledge is arguable. For this reason, we adopted the convention that any ambiguity that is driven by different views of the domain shall be apportioned to the domain knowledge dimension.

the length of a lap should be agreed. When the customer understood the issue, he specified that *if you swim in a swimming pool it [the device] should be able to understand when you switch direction*.

Domain knowledge component The domain knowledge component is further partitioned into three sub-components, to account for the different ambiguity phenomena that we experienced in practice. Hence, we found useful to refine the $acceptable^{D}_{a,i}(k)$ predicate as follows:

$$acceptable^{D}_{a,i}(k) = acceptable^{D_O}_{a,i}(k) \wedge acceptable^{D_C}_{a,i}(k) \wedge acceptable^{D_A}_{a,i}(k)$$

Here, D_O are the goals concerning the domain (i.e., its objectives), D_C are the rules of the domain (i.e., its constraints, in terms of regulations and business rules) and D_A is the mental application view that the analyst builds to figure out how the rules are applied in the real world. It is worth noting that these three dimensions are not the *actual* goals, rules or real-world views belonging to the domain, but those that are understood or inferred by the analyst during the interview, or according to his/her previous domain knowledge. In other terms, they form the vision of the domain from the perspective of the analyst. Of course, this vision might be wrong or, more precisely, not aligned with the vision of the domain of the customer, as we see by exemplifying the cases in which we have $\neg acceptable^{D}_{a,i}(k)$.

Example 3.8 ($\neg acceptable^{D_O}_{a,i}(k)$) Similarly to the cases in which we can have $\neg acceptable^{G}_{a,i}(k)$, these situations occur when the analyst does not understand the rationale or objective of a domain-related statement of the customer. For example, one of our domain experts works for a real-estate appraisal company, and she is in charge of establishing the market value of private properties. While explaining the process of providing an evaluation for a property, she said that she uses the data of neighboring properties with similar use (e.g., commercial, residential) and comparable size. She said that these data include both the selling price of the neighboring properties and the prices of the transactions of letting (i.e., rental contracts). Specifically, she said: *Besides the [sales and purchase] transactions, we need to consider also the transactions of letting*. Since, at that point of the conversation, the analyst had understood that the objective of the domain expert's work was estimating the selling value of properties, he could not understand the *goal* of having the rental price of neighboring properties. The expert explained that the objective of her work was not only giving a selling price, but also evaluating the potential revenue that the property could generate through rents. Then, the analyst understood that selling prices and rental prices contribute to the estimation of the market value of a property. In this sense, this situation led to understand a previous unknown business rule of the domain of the customer.

It is worth mentioning that the goal of the system to be developed was to provide financial information about the neighboring properties of the property that our customer was required to evaluate. Hence, the goal of estimating the market value of the property—discussed in this example—was a goal inherent to the domain, to which the system was required to contribute, but that requires other sub-goals to be addressed (e.g., performing the actual evaluation), which are outside the scope of the system.

Example 3.9 ($\neg acceptable^{D_C}_{a,i}(k)$) These situations occur when a statement of the customer is contrasting with the set of domain rules currently understood or inferred by the analyst. One example is the case of Example 3.6, in which the domain rules assumed by the analyst were contrasting with the rules—in this case, the *regulations*—considered by the customer. Let us give an additional example, in which the rule involved is not a regulation, but a business rule. One of our domain experts is a mechanical engineer working for a public research institution. He wishes to develop a system to support the purchase of components for his mechanical devices. During the interview, he expressed the following business rule: *When I know which component I need, I search who is selling that product [...], to receive an offer for the product*. This is a business rule, in the sense that gives a description of a dynamic behavior within the domain (we recall that business rules can also constrain the structure of the domain, see Sect. 2.3). From the previous statement, the analyst inferred that the customer was asking an offer for one single component—this can be regarded as an *inferred* business rule. Afterward, the customer described the offers that he receives from the sellers: *Based on the number of components, the seller specifies the cost*. This was contrasting with the idea that the customer was asking offers for one component only. Afterward, the customer clarified that, when he asks an offer, he also specifies the number of components of the same type that he wishes to order.

Example 3.10 ($\neg acceptable^{D_A}_{a,i}(k)$) These situations occur when the customer expresses a statement that is in contrast to the mental application scenario of the analyst. In our experience, this happens when the customer expresses a domain rule for which the practical applicability is unclear. For example, one of our customers is an expert in statistics working for a hospital and described the following scenario. Patients go to their general practice doctor, and the doctor might prescribe them a checkup. The hospital has an office that a patient can call to reserve the checkup. The

Table 1 Summary of ambiguity phenomena

k		k'		Type
$\neg interpretable(k)$	$\neg acceptable(k)$	$\neg interpretable(k')$	–	int. unc.
$interpretable(k)$	$\neg acceptable(k)$	$\neg interpretable(k')$	–	acc. unc.
$interpretable(k)$	$acceptable(k)$	$interpretable(k')$	$acceptable(k')$	mul. und.
			$\neg acceptable(k')$	cor. dis.
–	$\neg acceptable(k)$	$interpretable(k')$	$acceptable(k')$	u-inc. dis.
			$\neg acceptable(k')$	d-inc. dis.

int. unc.: *interpretation unclarity*; acc. unc.: *acceptance unclarity*; mul. und.: *multiple understanding*; cor. dis.: *correct disambiguation*; u-inc. dis.: *undetected incorrect disambiguation*; d-inc. dis.: *detected incorrect disambiguation*

domain expert said that the patient gets the number of the office *from the website of the hospital*. The analyst imagined a practical scenario. Since there are several hospitals, he could not understand how could the patient know which specific hospital's website to consult, and asked clarifications. The expert explained that *The [general practice] doctor suggests a medical specialist [for the checkup] and indicates which hospital should you call*.

3.4 Ambiguity

Given the previous definitions of the predicates $articulated_{c,i}(k)$, $interpretable_{a,i}(k)$ and $acceptable_{a,i}(k)$, we can give a more formal definition of ambiguity that accounts for all the cases perceived in our interviews.

Let k', with $k \neq k'$, denote any piece of information that can potentially be accessed by the analyst. An ambiguity occurs in the articulation of a unit of information k when:

$$\begin{aligned}\textbf{ambiguous}_i(k) = {} & articulated_{c,i}(k) \wedge \neg articulated_{c,i}(k') \\ & \wedge (\neg accessible_{a,i}(k) \vee interpretable_{a,i}(k'))\end{aligned}$$

The definition implies that the customer articulated a unit of information through a speech fragment and did not mean to articulate any other unit of information. However, the analyst either was not able to access this unit of information, or he/she interpreted the speech fragment of the customer in a way that was different from the intended meaning of the customer. This definition mimics the informal definition of ambiguity given in Sect. 2.1.

From this definition, we can derive six different feasible main classes. Indeed, by unfolding on the OR part of the condition (i.e., $\neg accessible_{a,i}(k) \vee interpretable_{a,i}(k)$), we have that an ambiguity occurs whenever:

$$\begin{aligned}\neg interpretable_{a,i}(k) \vee {} & \neg acceptable_{a,i}(k) \\ & \vee interpretable_{a,i}(k')\end{aligned}$$

That is, whenever the speech fragment is not interpretable, is not acceptable, or can be interpreted in a way different from the intended meaning of the customer. These classes are summarized in Table 1. The table considers only the feasible classes, since we discard the combination in which we have $acceptable_{a,i}(k) \wedge \neg interpretable_{a,i}(k)$.

4 Categories of ambiguities

This section discusses the categories of ambiguities derived from the formal definition introduced in the previous section. We classified them into four main classes, namely *unclarity*, *multiple understanding*, *incorrect disambiguation* and *correct disambiguation*, and in the remainder of this section, we present them accordingly. Furthermore, at the end of the section, we present a quantitative view on the different types of ambiguity that we identified.

4.1 Unclarity

The *unclarity* class includes situations in which the requirements analyst cannot give any interpretation or acceptable meaning to the unit of information expressed. This can happen because the information was not articulated in clear language, or for the usage of domain jargon, or because the interpretation is not acceptable in the mental framework of the analyst. This type of ambiguity can be formally represented as:

$$\begin{aligned}& articulated_{c,i}(k) \wedge \neg articulated_{c,i}(k') \\ & \qquad \wedge \neg accessible_{a,i}(k) \wedge \neg interpretable_{a,i}(k')\end{aligned}$$

If the speech fragment cannot be interpreted, we have $\neg interpretable_{a,i}(k)$, which causes the unclarity. In these situations, exemplified in Examples 3.1, 3.2 and 3.3, we speak about *interpretation unclarity*.

Moreover, depending on the component of the current mental framework that causes $\neg accessible_{a,i}(k)$, we can have different cases of unclarity situations. Examples of these cases were presented from Examples 3.4 to 3.10. In all the cases in which we have $\neg acceptable_{a,i}(k)$, we speak about *acceptance unclarity*.

4.2 Multiple understanding

The *multiple understanding* class includes situations in which the requirements analyst is able to give multiple acceptable interpretations to the expression of the customer, one correct and the other(s) incorrect, and is therefore left with the question of whether the intended meaning is the former or the latter(s). The formal representation of multiple understanding is:

$$articulated_{c,i}(k) \wedge \neg articulated_{c,i}(k') \wedge accessible_{a,i}(k) \wedge accessible_{a,i}(k')$$

Several examples of multiple understanding situations were experienced in our interviews, for example:

Example 4.1 (multiple understanding) One of our customers wanted to define a Web-based platform in which the citizens can send suggestions for laws to the parliament. The customer said that the platform was required to have *A dashboard to show [to the representatives of the parliament] what's going on in specific areas*. For the analyst, the term *areas* could mean geographical or thematic areas. Therefore, he asked the customer to which type of area was he referring, and the customer answered: *Geographical areas*.

We have noticed that in some cases the customer articulates an idea that has multiple meanings for the analyst, and each one is valid. Though not evident from the formalization, this situation is accounted in our definition of multiple understanding. An example might help clarifying these situations.

Example 4.2 (multiple understanding) Consider again the Web-based platform case. The customer stated that *The application should be connected to a social network*. Two meanings could be assigned by the analyst to this expression: (1) The application should have an embedded social network (we will refer to this need as k_1) and (2) the application should be connected with existing social networks (we will refer to this need as k_2). When asked which of the meaning was correct, or if both were correct, the customer said: *both*. Therefore, the k expressed by the customer was including both ideas ($k = k_1 \wedge k_2$). On the other hand, the k' understood by the analyst was considering also the case in which the two ideas could be exclusive ($k' = k_1 \veebar k_2$, in which $\veebar$ is the logical XOR operator). Therefore, also this situation falls in the category of multiple understanding.

4.3 Incorrect disambiguation

The *incorrect disambiguation* class includes situations in which the requirements analyst assigns a *single* interpretation to the expression of the customer, but this interpretation is different from the meaning intended by the customer. The formal representation of incorrect disambiguation is:

$$articulated_{c,i}(k) \wedge \neg articulated_{c,i}(k') \wedge \neg accessible_{a,i}(k) \wedge interpretable_{a,i}(k')$$

While performing requirements elicitation, this class of ambiguity normally includes *subconscious disambiguation* phenomena [10], which are hard to identify, unless the requirements analyst suspects that his/her interpretation of the expression of the customer is not correct. This phenomenon occurs when (1) the analyst can give an interpretation to the speech fragment of the customer, (2) this interpretation does not match with k (i.e., $interpretable_{a,i}(k')$) and (3) this interpretation is *not acceptable* in the mental framework of the analyst. We will specifically refer to these phenomena as *detected incorrect disambiguation*. The following example is representative of this category.

Example 4.3 (detected incorrect disambiguation) One of our customers wanted to build a *Fitness Tamagochi*, a game in which an avatar grows depending on how much workout the user does. The customer said: *It would be better if you could choose what type of character you want to create*. The analyst interpreted the verb *create* as *select when you start the game*. However, he could not understand the goal of having different characters to be selected (i.e., $\neg acceptable^{G}_{a,i}(k')$). Therefore he asked: *So, you can choose the character?* The customer replied: *Actually you cannot [...] you can possibly become [a specific character]*. After discussing with the customer, it became clear that he wanted the avatar to have different transformations depending on the type of training performed and that with the term *create*, she basically intended "become."

In the other cases, in which k' is both interpretable and acceptable, nocuous subconscious disambiguation phenomena [10] are likely to occur. Consider as example the case in which the customer is articulating a requirement and is using a domain-specific term that has a meaning for the analyst, and is also acceptable in his/her current mental framework. In this case, the analyst will include the new requirement in the set of requirements currently elicited, but with an incorrect interpretation, which might or might not emerge after the requirement is committed to a requirement document. We refer to these cases as *undetected incorrect disambiguation*. In our interviews, we were able to see these phenomena thanks to further discussion with the customer along the interview, which revealed that, during the conversation, an undetected incorrect disambiguation occurred. A first example is

Example 3.5, in which an incorrect disambiguation phenomenon was later discovered thanks to an acceptance unclarity ($\neg acceptable^{R}_{a,i}(k)$). Another similar example is reported below.

Example 4.4 (undetected incorrect disambiguation) In one of our interviews, the customer wanted to have an automated baby swinger. She said: *I want something that can change. A component that relax her [the daughter] is that she feels the novelty in the movement*. The analyst interpreted this sentence as a change in frequency of the movement and did not ask for further clarification. However, during the conversation, and detailing the behavior of the system from the user interface point of view, the customer said: *You can choose different sequences of movement, three in this direction, two in this direction*. This sentence, together with the discussion that followed, clarified that the customer wanted something that changes in terms of direction, and not in terms of frequency. The undetected incorrect disambiguation was discovered thanks to an *acceptance unclarity*, since the speech fragment of the customer showed an inconsistency with respect to the requirements understood by the analyst (i.e., $\neg acceptable^{R}_{a,i}(k)$).

4.4 Correct disambiguation: also-known-as innocuous ambiguity

The *correct disambiguation* class includes situations in which the requirements analyst can assign more than one interpretation to the speech fragment of the customer, but the only interpretation that appears acceptable to the analyst is the one that matches the meaning intended by the customer. Formally:

$$articulated_{c,i}(k) \wedge \neg articulated_{c,i}(k') \wedge accessible_{a,i}(k) \wedge interpretable_{a,i}(k') \wedge \neg acceptable_{a,i}(k')$$

This is a theoretical phenomenon that we were not able to able capture in our interviews. Indeed, when a disambiguation of this type occurs, the interview continues without any interruption and without a conscious awareness that the disambiguation actually took place. We hypothesize that this is a subconscious disambiguation phenomenon, which happens any time a listener has to assign the most likely meaning to an expression, given a certain context. For example, if someone tells us: *I went to the bank to open an account*, we will never think that the term *bank* could refer to the part of land adjoining a river, since the occurrence of the term *account* in the context of the sentence helps us disambiguate the term *bank*. In other terms, the latter interpretation is not acceptable ($\neg acceptable^{D_A}_{a,i}(k')$).

Correct disambiguation is the basis of the so-called *innocuous ambiguities* observed by Chantree et al. [12] and Yang et al. [78]. In these works, which focus on specific syntactic ambiguities in NL requirements, namely coordination [12] and anaphoric [78], particular emphasis is given to the distinction between innocuous and nocuous ambiguities. The former include those cases in which a requirement having multiple syntax trees has a single reading in practice. The latter include those situations in which multiple readings are possible. Innocuous ambiguities can be regarded as correct disambiguation phenomena. Indeed, a theoretical ambiguity exists in a fragment such as *The size of vector-based inputs and outputs shall be defined* (adapted from the paper of Chantree et al. [12]), since multiple syntax trees are possible— i.e., *vector-based* can be referred to *input* only, or also to *output*. Nevertheless, given the semantic relation between input and output, a reader tends to associate *vector-based* to both *input* and *output* and does not perceive the potential ambiguity. Similarly, a theoretical lexical ambiguity [10] exists for the term *bank* in the previous example, since *bank* has multiple entries in a dictionary. However, this ambiguity is automatically resolved by the human brain as a correct disambiguation, and it becomes an innocuous ambiguity. From the commonsense point of view, it is *not* an ambiguity.

Hence, we have seen that our definition of ambiguity actually withholds the definition of the process of correct understanding. Does this disrupt our whole theory? No. The boundary between correct understanding and correct disambiguation is blurry, and we prefer our definition of ambiguity to enclose this fuzziness. Moreover, this definition embodies the notion that ambiguity occurs whenever we speak, even though we are not always aware of that. Notice that this is in line with the general terminology used in natural language processing, where disambiguation is used as a synonymous of understanding [41].

On the other hand, it is possible to amend our definition of ambiguity and define a concept of strict-ambiguity, by excluding the class of correct disambiguation. With such an amendment, the definition of ambiguity given in Sect. 3.4 becomes:

$$\begin{aligned}\mathbf{ambiguous-strict}_i(k) = {} & articulated_{c,i}(k) \wedge \neg articulated_{c,i}(k') \\ & \wedge (\neg accessible_{a,i}(k) \vee interpretable_{a,i}(k')) \\ & \wedge \neg(accessible_{a,i}(k) \wedge interpretable_{a,i}(k') \\ & \wedge \neg acceptable_{a,i}(k'))\end{aligned}$$

In the remainder of the paper, when we will speak about ambiguity, we will refer to this most recent notion, which includes only interpretation unclarity, acceptance unclarity, multiple understanding and incorrect disambiguation (both detected and undetected).

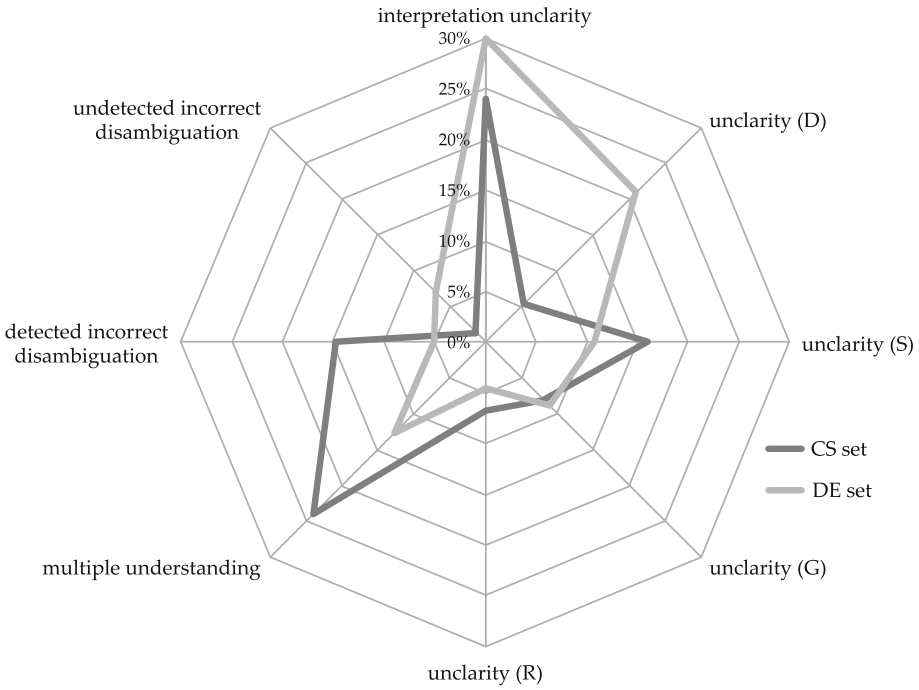

Fig. 2 Statistics concerning the different categories of ambiguity identified in our interviews

4.5 Quantitative view of ambiguity types

Though this research is not a rigorous empirical study, it is useful to present some quantitative aspects that emerged from our work. Indeed, even though we cannot draw general conclusions, the reader might benefit from some numerical values to have a clearer view of the baseline from which we developed our vision. In particular, it is interesting to look at the differences between interviews involving customers with a computer science background and interviews involving domain experts. We will refer to the former group of interviews as the CS set and to the latter as the DE set. Let us now look at the different categories of ambiguities identified for the two sets. Figure 2 summarizes the percentage values for each set. We see that interpretation unclarity and unclarity (D)[5]—i.e., acceptance unclarity due to the domain component—are the most frequent cases for the DE set, with 30 and 21 % of cases, respectively. A large part of the ambiguity cases with domain experts were actually due either to the usage of a domain terminology that was unknown to the analyst—leading to interpretation unclarity—or to conflicts with the domain view of the analyst—leading to unclarity (D). Interpretation unclarity is also frequent for the CS set (24 %), for which also multiple understanding cases appear to be dominant (24 %). In the CS set, also detected incorrect disambiguation phenomena (15 %) and unclarity (S) (16 %)—i.e., acceptance unclarity due to the specification component—are also common.

In a sense, we can say that interviews with computer scientists were characterized by a higher degree of pure ambiguities, i.e., incorrect disambiguation (mainly detected) and multiple understanding, while interviews with domain experts were characterized by a higher degree of unclarity.

One might also notice that the number of undetected incorrect disambiguation cases is higher for the DE set with respect to the CS set (7 vs. 1 %). We hypothesize that this phenomenon can be traced to two main causes. First, interviews with domain experts were, in average, longer, and this could have left more time to discover previously undetected incorrect disambiguation cases. Secondly, most of the interviews in the DE set were performed *after* interviews in the CS set, and, more importantly, after defining the first version of our framework [23]. Hence, we hypothesize that a higher awareness of the analyst of the potential of incorrect disambiguation might have led him to detect a higher number of them.

It is worth highlighting that, since the evaluation was performed by the authors, and since the framework emerged together with the analysis of the fragments, these data have to be considered as part of the vision presented, and not as an empirical confirmation of this vision. In other terms, the reader is advised not to draw general conclusions from these data. Instead, the interested requirements analyst or researcher is encouraged to refer to our framework to assess its validity, and possibly extend it according to his/her experience.

[5] We do not show the different sub-categories, to give evidence of the dominance of the domain component with respect to the other types of unclarity.

5 Disclosing tacit knowledge

Ambiguities in requirements elicitation interviews are strictly interwoven with the concept of *tacit knowledge* [25, 28, 48, 60, 75]. In the remainder this section, we will analyze this relationship and suggest how to use ambiguity as a tool to disclose tacit knowledge.

5.1 Ambiguity and tacit knowledge

To better understand the connection between ambiguity and tacit knowledge, we rely again on the framework for tacit knowledge illustrated by Gervasi et al. [28]. The framework considers four different classes of knowledge that play a role in requirements elicitation:

- *known known*, as relevant information that is successfully passed from the customer to the analyst;
- *known unknown*, as relevant information that was not expressed by the customer, but that the analyst knows or suspects that the customer has;
- *unknown known*—i.e., tacit knowledge—as relevant information that the customer can in principle express, but does not pass to the analyst, and the analyst is not aware of the existence of this information;
- *unknown unknown*, as relevant information that is unknown to both the customer and the analyst.

Sutcliffe and Sawyer [75] provide some examples of these phenomena. Here, we will contribute with examples taken from our direct experience.

Example 5.1 As an example of *known unknown*, consider the case of one of our interviews, in which the customer, a domain expert in Public Administration, wishes to realize a Web-based platform to monitor the activities of different European Union (EU)-funded projects. The analyst knows that the customer has the knowledge associated with the current process adopted for monitoring such projects, and will ask questions concerning the process. This sort of knowledge is part of the known unknown class.

Example 5.2 As an example of *unknown known*, consider again the case of the History of Arts expert. The customer knew that pictures of paintings are available in specialized archives called photo libraries, but, in many of the cases considered interesting for the customer, are in paper format and *not* digitalized. However, he did not consider this aspect relevant for the system, and this information was withheld from the analyst. The fact was discovered only when the analyst asked: *In which format do you have the pictures, .jpg, .tiff?* (assuming that photo libraries were storing digital pictures), and the customer—after asking whether the information was important—replied: *Well, a digital archive does not exist!*.

Example 5.3 As an example of *unknown unknown*, consider the following case, which involves regulatory requirements. In one of our interviews, we had a customer who wanted to define a smart elevator system. The conversation took into account many functional and safety aspects, but none of the participants raised issues concerning the certification of the system, since, in the moment of the interview, this aspect was unknown to both the customer and the analyst.

Ambiguity can turn a potentially *known known* into a *known unknown* in practice. In other terms, in the presence of ambiguity, information that can potentially pass from the customer to the analyst becomes information that is not successfully accessed by the analyst. Nevertheless, this information becomes accessible when the analyst detects the ambiguity during the articulation of the information, and asks the right questions to enlarge the space of shared understanding. In cases of undetected incorrect disambiguation, a potentially *known known* can even become an *unknown known*, i.e., a *tacit knowledge* that the analyst does not suspect it exists, and that, if no further event occurs, will not be articulated again by the customer.

Ambiguities are also a *resource*, since an ambiguity might appear in situations in which the customer cannot articulate an idea properly. In some cases, this might happen because some domain knowledge could be hidden in his/her mind in the form of procedural knowledge [46], which is sometimes hard to express. In other cases, when the unit of information refers to the system-to-be, an improper articulation might occur because the customer's needs are still vague to the customer himself or herself. These cases can contribute to reveal both *unknown unknown* situations—in case of vague needs—and *unknown known*—in case of procedural knowledge—and turn them into *known unknown*. Then, the analyst can leverage the ambiguity to ask for further clarifications and possibly opening new directions for scoping the problem.

5.2 Disclosing tacit knowledge through ambiguity

We have previously introduced two predicates defined by Gervasi et al. [28], namely $articulated_{c,i}(k)$ and $accessible_{a,i}(k)$ (see Sect. 3). The framework of Gervasi et al. [28] introduces two additional predicates to define tacit knowledge, which are:

- $expressible_{c,i}(k)$: k can be expressed through i by the customer c;
- $relevant(k)$: k is relevant for the system or project that is discussed in the interview.

Moreover, the framework considers also the predicate $accessible_{c,i}(k)$, to express the idea that a unit of

information is accessible by the customer. The framework illustrates several interactions of the given predicates and defines the different notions of tacit knowledge (i.e., when an information is not expressible by a customer, or when it is not accessible for the analyst). Here, we will refer to the following notion of tacit knowledge:

$$\begin{aligned}\textbf{tacit}(k^*) = {} & relevant(k^*) \wedge accessible_{c,i}(k^*) \\ & \wedge \neg expressible_{c,i}(k^*) \wedge articulated_{c,i}(k) \\ & \wedge \neg accessible_{a,i}(k^*)\end{aligned}$$

The expression means that the customer has access to a relevant unit of information k^* but cannot express it properly, or in its entirety, through i. Therefore, he/she articulates a unit of information k that has some link in his/her mind with k^*. The inadequate or insufficient expression of the customer makes k^* not accessible for the analyst within the expression i. Let us now consider the examples of ambiguity presented in this paper in light of this notion.

Consider the example Example 3.1, in which the History of Arts expert mentioned the *connoisseurship method*. The customer articulated a unit of information k that was linked to a larger topic k^*—i.e., the procedure associated with the method—which was not accessible by the analyst. However, the *interpretation unclarity* perceived by the analyst helped in discovering the k^* topic and, ultimately, achieving $accessible_{a,i}(k^*)$. Instead, in the other example of interpretation unclarity (Example 3.2), the customer could not express his idea k^*—i.e., preserving the shades of the wall—properly, and used a vague expression. The identification of the problem helped the analyst in accessing the tacit idea k^* of the customer. A similar case of vague expression can be observed in Example 4.3, in which a *detected incorrect disambiguation* case occurred, and in Example 4.1, in which a *multiple understanding* situation is shown.

Another representative case is Example 3.4, in which the hidden goal of the bus tracking system—i.e., being able to change transportation means—can be regarded as tacit knowledge k^*, which the analyst elicited after perceiving an *acceptance unclarity* for the information k—i.e., the requirement concerning the need to know the position of the bus. Still on acceptance unclarity, Example 3.6 shows that the requirement of the customer was associated with a hidden domain knowledge k^*—i.e., the fact that in certain municipalities people who do not recycle properly get fined. This knowledge was unknown to the analyst and was discovered after clarification of the ambiguity.

The analysis of these examples should give the rationale under which we perform our statement that ambiguity can help in discovering tacit knowledge, related either to requirements (Examples 3.2, 4.3, 4.1), to goals (Example 3.4) or to the domain (Examples 3.6, 3.1).

Disclosing unknown unknowns In some cases, ambiguity also led to disclose *unknown unknowns*. An example in this group concerns the swim keeper in Example 3.7, in which a previously unknown unknown requirement (i.e., the need to detect when the swimmer changes direction) emerged thanks to an *acceptance unclarity*. Unknown unknown can be associated also with domain aspects or even with goals. In our experience, the identification of domain unknown unknowns occurred mainly (a) in the case of regulatory requirements, as in Example 5.3, or (b) when the knowledge of some domain rule appears to be needed to define the system, but the customer does not know that rule.[6] In both cases, another source, i.e., a stakeholder or a document, has to be identified in the domain to access the required information. Instead, goal unknown unknowns can be discovered when the customer expresses a novel requirement for the system, or describes a practical domain scenario, which is inconsistent with the system goals previously specified, causing an acceptance unclarity (i.e., $\neg acceptable^{G}_{a,i}(k)$). Let us see these phenomena in two examples coming from our interviews.

Example 5.4 (domain unknown unknown) Consider again the mechanical engineer of Example 3.10. The customer said that a recent norm asks him to provide his institution with evidence that the component that he buys is actually the most convenient in the market. He said: *[According to the norm,] we have to search all the sellers of that specific component [to ask for an offer]*. According to his view of the domain, the analyst considered the task of knowing *all* the sellers unfeasible in practice (i.e., $\neg acceptable^{D_A}_{a,i}(k)$), and he asked clarifications. The customer said: *Well, the norm is changing [...]*. After some discussion, it became clear that the customer did not know the actual norm. Hence, to develop a system to automatically support the order of components, and that would comply to the norm, another stakeholder with experience about the norm should have been involved, or, more rigorously, the actual norm should have been retrieved.

Example 5.5 (goal unknown unknown) Consider the physician of Example 3.3, who wishes to develop a system to predict the dermal or systemic pathology of a patient, given his/her symptoms. The customer said that the system would have addressed the following problem. Many general practice doctors tend to send patients to the dermatologist when the patient has cutaneous manifestations, because in many cases they cannot precisely associate a disease to the manifestation. This leads to many unnecessary visits to the specialist. Indeed, he said: *The main problem is that the non-specialist doctor [...], when a*

[6] This latter case is only speculative, since we were not able to see these cases in practice.

patient arrives who has cutaneous manifestations, he [the doctor] sends him [the patient] to the dermatologist. Addressing this problem was assumed as a domain goal by the analyst (i.e., it became part of the D_O component), and explicitly agreed as a system goal, becoming part of the G component. To have more insight, the analyst asked which were the most frequent cases in which the general practice doctor sends the patient to the dermatologist unnecessarily. He started exemplifying, and then he said: *Often I see people to whom the [general practice] doctor prescribed an antihistamine, and instead he should have prescribed an antibiotic*. So the problem was also that some incorrect diagnosis was performed by general practice doctors. This fragment was inconsistent with the previously described problem (i.e., $\neg acceptable^{D_O}_{a,i}(k)$) and in turn caused an inconsistency with the main goal of the system that was previously agreed (i.e., $\neg acceptable^{G}_{a,i}(k)$). The analyst made this inconsistency explicit, and, after ten seconds of hesitation, the customer said: *We can say that the excessive number of visits to the dermatologist is not the main objective, the main objective is making the general practice doctor able to recognize whether a certain pathology could be severe*. This latter objective can be regarded as an unknown unknown goal, disclosed thanks to an acceptance unclarity.

Disclosing tacit knowledge The only cases in which tacit knowledge cannot be accessed, are those of *undetected incorrect disambiguation*. Indeed, except for those situations in which another ambiguity helps in discovering the problem (see Examples 3.5 and 4.4), if an undetected incorrect disambiguation occurs, the analyst has no means to discover tacit knowledge during the interview.

Therefore, we can state that tacit knowledge k^* can be potentially disclosed when there is an ambiguity in the conversation with the customer, but no *undetected incorrect disambiguation* phenomena occurred. Formally:

$$\mathbf{D-tacit}(k^*) = \mathbf{tacit}(k^*) \wedge \mathbf{ambiguous-strict}_i(k) \wedge \neg(\neg acceptable_{a,i}(k) \wedge accessible_{a,i}(k'))$$

Of course, this finding relies on the need to have $articulated_{c,i}(k)$, with k associated with some tacit knowledge k^*. Hence, those cases in which the customer does not articulate any unit of information that might be linked to some tacit knowledge cannot be accessed through ambiguity, and different strategies are required to address these situations [75].

5.3 Quantitative view of tacit knowledge

According to the rationale explained above, we inspected the fragments that caused ambiguity in our interviews, and, through joint discussions among the authors, we considered whether the identified cases helped in disclosing tacit knowledge. The result of this analysis is shown in Fig. 3 for the CS set and in Fig. 4 for the DE set. A first evident difference among the two sets resides on the dominant type of tacit knowledge discovered. In interviews with computer scientists, most of the tacit knowledge discovered through ambiguity is related to requirements (59 %), i.e., some requirement was incorrectly assumed by the customer as *understood* in the context of the conversation. Instead, in interviews with domain expert most of the tacit knowledge was associated with the domain (61 %). This is a predictable result, also looking at the different types of ambiguities found during the interviews, and reported in Fig. 2. Indeed, in the DE set, the higher number of acceptance unclarities associated with the domain knowledge dimension led to the disclosure of a higher amount of domain tacit knowledge. We also notice that tacit

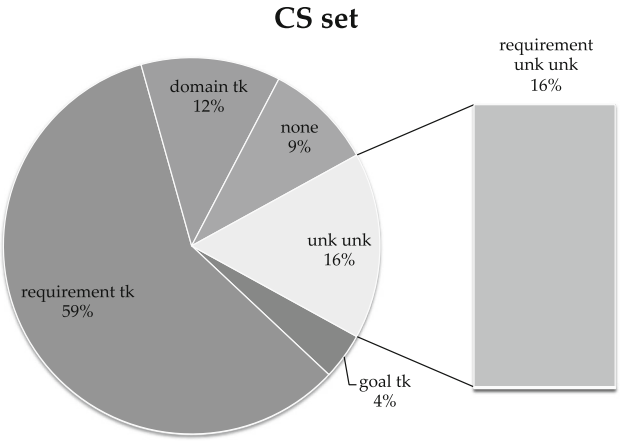

Fig. 3 Types of knowledge discovered during the interviews with computer scientists (tk = tacit knowledge, unk unk = unknown unknown)

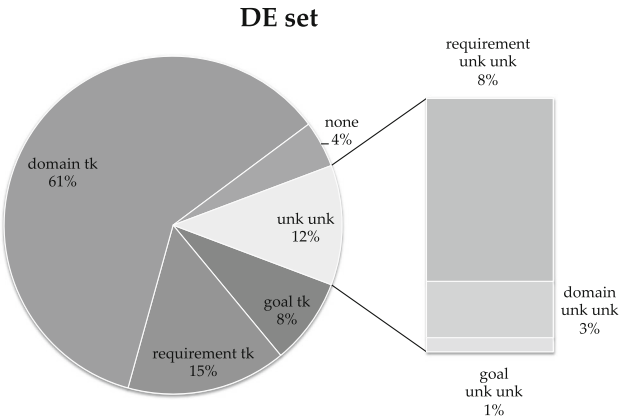

Fig. 4 Types of knowledge discovered during the interviews with domain experts (tk = tacit knowledge, unk unk = unknown unknown)

knowledge about goals is rather low for both sets (4 % for the CS set and 8 % for the DE set). We argue that these values could be possibly improved by involving a business analyst in the interviews, who is in principle more aware of the high-level needs and motivations of a customer.

Concerning the unknown unknown class, we see that, in both sets, a non-negligible number of cases led to the disclosure of some unknown unknown. But while in the CS set all cases are associated with unknown unknown requirements that emerged during the conversation, in the DE set we also observed a small part of unknown unknowns related to domain and goals. The major number of unknown unknown requirements in the CS set can be traced to the different types of systems discussed in the interviews. Indeed, in the DE set, systems were more domain specific, i.e., solutions to particular problems of the domain (see, e.g., Examples 3.3 or 3.9). Instead, in the CS set, the systems discussed were more innovative kinds of consumer's products (see, e.g., Examples 3.2 or 3.5). Hence, unknown unknown requirements are probably due to the innovative aspect of the products discussed: Neither the customer nor the analyst knew in advance the potential requirements that the system would need. On the other hand, unknown unknown domain aspects and goals probably remained uncovered also in the CS set, due to the lower awareness of the analyst of the issue of tacit knowledge. Indeed, recall that the interviews in the CS set were completed before developing the theory reported earlier [23]. Of course, given the low percentages of these categories of unknown unknowns in the DE set, also sheer chance could have played a role.

We see that only in 9 and 4 % of the cases, the ambiguity did not lead to disclosing some type of knowledge. These were the cases in which the ambiguity either was not resolved, was caused by babbling [28], or was the result of inapplicable technological expectations of the customer (i.e., the customer had a solution in mind, but the solution was unfeasible from the technological point of view). It is worth highlighting that, since the evaluation was performed by the authors, these results have to be considered as part of the vision presented, and not as an empirical confirmation of this vision.

6 Discussion

Our phenomenology of ambiguity and the analysis on how ambiguity is a means to disclose tacit knowledge represent a first step in much broader research on the role of ambiguities in interviews. Such research poses fundamental challenges and requires a systematic validation of the corresponding answers. In the following, we analyze four main challenges and some lesson learnt about them during our on-the-field experience.

6.1 Challenge 1: Identification of ambiguity cues

We have shown that ambiguity can be used to discover tacit knowledge, but *how can an analyst discover ambiguity?* In some cases, the detection of typical term-based cues discussed in the previous literature on NL requirements (see e.g., [9, 10, 33, 44]) can provide a first help, as shown in Example 3.2, in which the vague expression *taking light into account* triggered the discovery of tacit requirements knowledge. In other cases, domain-specific terms that are unknown to the analyst (e.g., *connoisseurship method* in Example 3.1) can also be used as cues. However, in these cases, it is important for the analyst to identify when a *common* term is used with a *specific* meaning in the domain of the customer. For example, our Public Administration expert was using the common term *program* to refer to his administrative office. Techniques have been developed to detect domain-specific terms and abstractions in NL requirements (see, e.g., [20, 26]). At the conversation level, analogous goals should be achieved also for oral interviews.

Term-based cues can also be associated with a peculiar phenomenon of requirements interviews that we observed and that we call *meaning migration*. When a novel system has to be developed, novel terminology is needed to describe its structure and behavior. Therefore, the customers tend to use common terms to denote system-specific entities or actions. The term *tap* in Example 3.5 is an example of this phenomenon, in which a term *migrates* from the vocabulary of the customer to the terminology of the system. Identifying this type of cues is not easy and requires further research. From our experience, we hypothesize that an index that computes the degree of fitness of a term within a given linguistic context (i.e., an indicator that tells how common is the usage of the term in conjunction with the other terms used) can quantify the potential novelty—and, hence, ambiguity—of the term within the context. For example, we expect that the term *tap* is not often used in a linguistic contexts that speak about windshield wipers.

In many cases, ambiguity cannot be directly associated with the usage of specific terms (see, e.g., Examples 3.6 and 3.7), and research has to go beyond term-based cues. In these situations, pragmatic cues at the level of discourse have to be identified, which take into account the pragmatic facet described in Sect. 2.3. In NL requirements, we studied methods to detect pragmatic ambiguities [21, 22] that consider the background knowledge of a reader. However, the research in this domain is still at its early stages, and further insights are required to accounts for all the components of the pragmatic facet, their evolution along the interview and their connection with the discourse-level structure of customer–analyst dialogs [15].

6.2 Challenge 2: Ambiguity-based elicitation methodologies

To profitably employ ambiguity in interviews, elicitation methods that leverage ambiguities have to be defined and integrated with existing practices [62, 75]. Such methods should take into account the fact that the analyst has to be trained in ambiguity detection, keep his/her ears open to ambiguity cues (Challenge 1), and be able to explicitly discuss his/her current understanding of the problem domain. A good practice that we used was taking notes of ambiguous terms used by the customer during the conversation. Such terms were used to ask further clarification to the customer, after the customer terminated his/her discourse. Another good practice was immediately summarizing the discourse to the customer. Rephrasing is indeed recognized as a powerful strategy to achieve a shared understanding [43, 68]. In our experiments, we were able to detect cases of incorrect disambiguation specifically thanks to rephrasing. Moreover, we saw that summarizing the discourse helps establish a shared terminology and gently drives the customer toward an algorithmic mindset and a lexicon, in which terms have precise meanings, and system goals and behaviors are sharply defined. In particular, during the interviews with the History of Arts and Public Administration expert, the summarization let a common vocabulary [53] gradually emerge, a vocabulary whose terms came from the domain of the customer, but whose usage structure was more computer science oriented.

6.3 Challenge 3: Ambiguity in the process

Requirements collected during elicitation evolve during the whole software lifecycle. After the requirements have been elicited, they are normally committed to a document, most of the time in NL [51]. In a NL requirements document, the terminology and the syntax tend to be more precise, and ambiguity not found during elicitation could emerge in the editing phase, when the requirements analyst discovers that some concepts that emerged during elicitation cannot be properly expressed in the written language. On the other hand, the precision could also lead to over-confidence of the requirements analyst as to the absence of ambiguity, whose presence could be hidden by the more formal surface of the written text. Understanding how the concept of ambiguity *evolves* along the process is therefore paramount to define proper tools and techniques to detect ambiguities in different stages of development. To this end, a phenomenology of ambiguity, similar to the one presented in this paper, should be defined at the levels of editing, negotiating, refining and testing requirements. Such a phenomenology should take into account the expected readers of the requirements, and the components of the pragmatic dimension involved in each phase. For example, in the testing phase, the reader of a requirement will be a software testing expert, who is more concerned with input–output relations than with goals, which are more relevant during the elicitation phase. Therefore, the concept of *acceptance unclarity* has to be tailored for these cases, to account for different dominant pragmatic components of the reader's mental context.

6.4 Challenge 4: Ambiguity on the customer's side

In our framework, we have focused on ambiguities perceived by the analyst, but to have a complete view of the phenomenon, also the perception of ambiguity from the customer side has to be investigated and modeled. In our experience, for the analyst, many ambiguities are triggered by domain-specific terms, while for the customer, many ambiguities are triggered by technical terms or computer science jargon, e.g., acronyms such as "*.jpg," "FTP," "SVM," and terms such as "Actor," "Scenario," "Model." Achieving a shared understanding [32] is a primary goal in interviews. Just as analyst-perceived ambiguity can disclose tacit knowledge of the customer, we hypothesize that customer-perceived ambiguity can disclose the tacit knowledge of the analyst. Such tacit knowledge—which includes technological and implementation aspects—should become part of the common ground [14] between customer and analyst. Hence, we argue that a correct comprehension of the customer's side of ambiguity can be used to enlarge the space of shared understanding of the problem domain. However, since we analysts are aware of ambiguities in general and our jargon, we can prepare upfront a lexicon of technical terms, acronyms and jargon. After understanding the customer's side of ambiguity, we expect to have a sufficiently clear view of both sides to start investigating the emergent properties of their interaction, and the role of dialog and argumentation.

7 Related works

The current paper stems from previous works in three main fields, namely ambiguity in NL requirements, requirements elicitation interviews and tacit knowledge, and identifies a red thread that crosses these three research areas. Therefore, we present relevant related works within these areas, and we make our contribution explicit with respect to previous literature.

7.1 Definitions and classifications of ambiguity

Ambiguity has interested philosophers and linguists for very long time. One of the first studies is included in the

Sophistical Refutations of Aristotle [4]. This philosophical study classifies ambiguity into three categories: the one that occurs when a used expression (or name) has strictly more than one meaning; the one that occurs when an expression is used on purpose to have more than one meaning; and the one that occurs when words that have a simple sense taken alone have more than one meaning in combination. Over the years, linguists and philosophers have kept studying the phenomenon of ambiguity and created detailed ambiguity classifications (see Sennet [70] for a reference work in linguistics, and Empson [19], for a reference work in literary criticism, originally published in 1930). Since natural language (NL), which is inherently ambiguous and imprecise [7], is often involved in the requirements life cycle, the phenomenon of ambiguity has been deeply analyzed also by the requirements engineering (RE) community. In particular, the literature focuses on the analysis of ambiguity in requirements documents, i.e., written requirements. To the best of our knowledge, our work—originally presented at RE'15 [23]—is the first attempt to analyze how ambiguity appears in requirements elicitation interviews. Let us know look at the different definitions and classification of ambiguity in RE.

7.1.1 Definition of ambiguity

In requirements engineering, there is not yet a single, comprehensive, accepted definition for ambiguity [10]. For example, according to the IEEE Std 830-1998 [73], a requirement is ambiguous if it admits more than one interpretation—i.e., to be unambiguous, a requirement needs to have a unique interpretation. This is a rather strict definition, since it entails the idea that *any* subject shall give the same interpretation to the requirement. With a more pragmatic perspective, Harwell et al. [37] define a requirement as unambiguous if different stakeholders with similar backgrounds give the same interpretation to it. Hence, if a lack of detail in a requirement can be covered by the shared background knowledge of the stakeholders, then the requirement is not ambiguous. Similarly, if a requirement has multiple syntactic readings, but all that stakeholders read it in the same way, the requirement is not ambiguous. In a sense, this definition is related to the notion of innocuous ambiguity, introduced by Chantree et al. [12], and referred in Sect. 4.4.

Gause and Weinberg [27] implicitly introduce the role of the information producer in their definition of ambiguity. Indeed, they define requirements ambiguity in relation to its sources, namely, missing information and communication errors. The former can be caused by humans errors in observation and recall, tendency to leave out self-evident facts, and incorrect generalization. The latter are usually due to expression inadequacies in the writing. Similarly, Schneider et al. [69] define as ambiguous any important term, phrase, or sentence that is *essential* to understand a system behavior, and that (a) is left undefined, or (b) is defined in a way that can cause confusion and misunderstanding. As for Gause and Weinberg's [27], this definition tells that missing information and communication errors are sources of ambiguity. However, the definition focuses solely on those communication items that carry essential information. In this sense, anything that is not considered essential is not ambiguous.

All these different definitions are interesting and give a flavor of how researchers and practitioners have tried to constrain the fluid notion of ambiguity. A throughout analysis of the different definitions is given by Berry and Kamsties [7]. In terms of balance between rigor and pragmatism, our definition in Sect. 2.1 is somehow in between that of the IEEE Std 830-1998 [73] and that of Harwell's et. al. [37]. Moreover, as did Gause and Weinberg [27] and Schneider et al. [69], we introduce the role of the information producer. Our idea is that, given an expression, a correct interpretation exists in the intention of the producer of the expression, i.e., the customer in our case. However, the interpretation needs to be acknowledged by the receiver of the information, i.e., the analyst. Differently from Aristotle's definitions [4], our definition does not explicitly account for intentional ambiguities.

7.1.2 Categorization of ambiguity

Now, it is useful to look at the *categorizations* of requirements ambiguity provided in the literature. We will discuss two reference works, the traditional categorization of Berry et al. [10], and one, more recent, provided by Massey et al. [49].

Berry et al. [7, 10] distinguish ambiguity into four categories: *lexical* (i.e., the terms used have several meanings), *syntactic* (i.e., the requirement sentence has more than one syntax tree, each one with a different meaning), *semantic* (i.e., the sentence has more than one meaning within its context) and *pragmatic* (i.e., the meaning of the sentence depends on the context in which it is used). Moreover, they speak about two other phenomena that are closely related to ambiguity, namely *vagueness* (i.e., a term is vague if it admits borderline cases, for which no inquiry or conceptual analysis can assign a precise meaning to the term) and *generality* (i.e., the expression could be specified more precisely).

Massey et al. [49] define as unambiguous those statements with only a single, clear interpretation and specify that this includes statements with no interpretation, such as vague or incomplete statements. This latter amendment is in line with our vision, which includes cases in which the analyst cannot give any interpretation. Starting from this

definition, Massey et al. [49] propose a taxonomy with six ambiguity types, namely *lexical*, *syntactic*, *semantic*, *vagueness*, *incompleteness* (i.e., when relevant details are missing) and *referential* (i.e., when a word or phrase does not have a clear reference). Lexical, syntactic and semantic ambiguity—and vagueness as well—are defined as in the work of Berry et al. [10]. However, for Berry et al. [10], vagueness is treated as a phenomenon that is distinct from ambiguity. Moreover, to our understanding, generality of terms is treated as part of lexical ambiguity by Massey et al. [49], while generality of sentences could be associated with the definition of incompleteness. Finally, referential ambiguity according to Massey et al. [49] is considered as a sub-type of pragmatic ambiguity by Berry et al. [10], a category that is not explicitly mentioned by Massey et al. [49]. Of course, this comparison is based on our understanding of the works and might be questionable.

The main difference between these categorizations and ours is that Berry et al. [10] and Massey et al. [49] mainly focus on the triggers—or *cues*—of ambiguity, while we investigate the perceptual consequences of ambiguity from the point of view of the analyst. Hence, our categorization shall be seen as complementary to the previous ones and as a means to better understand the phenomenon. A throughout work on ambiguity cues in interviews, as foreseen in Sect. 6, will clarify the relation between the mentioned classifications and ours.

7.2 Preventing and detecting ambiguities

Several techniques have been developed to address the problem of ambiguity in written requirements. Such techniques can be broadly partitioned into two sets. The first set of techniques suggests to use formal, semiformal languages or constrained NL to prevent or limit ambiguity. The second set of techniques starts from unconstrained NL and generally aims at detecting ambiguity.

7.2.1 Constrained NL and (semi-)formal languages

A requirements document can be written in NL, or using a formal or semiformal language. Both the approaches have advantages and disadvantages [6]. In particular, NL has the advantages that does not require technical competences to be used and is always more or less understood by all stakeholders. The main disadvantage is of course the "more or less," i.e., ambiguity. Conversely, (semi-)formal languages are inherently unambiguous, but they require time to be learnt and most stakeholders cannot understand them. The debate on which of the two approaches is more appropriate is still open. However, through an analysis of two empirical studies, Kamsties [42] concludes that formalization does not help to eliminate ambiguity from informal requirements documents. Indeed, during the formalization process the analyst makes implicit assumptions, transforming ambiguities into errors. Analogously, Osborne [58] shows that, partly because of their restrictiveness and partly because they are difficult to use, formal languages do not solve the ambiguity problem. As an alternative approach, he proposes the use of a controlled natural language, which does not solve all the ambiguity problem but is easier to be analyzed. Following the same idea of constraining NL, Ambriola et al. propose the CIRCE framework [3], which encourages a rigorous use of NL. The idea is to structure requirements documents in the three layers of designations, definitions and requirements and to apply parsing and information extraction techniques to obtain precise and unambiguous information from NL requirements. Again based on the idea of using a controlled NL, but on a complete different note, Zowghi et al. [82] and Gervasi and Zowghi [29] suggest logic as a tool to identify and analyze inconsistency in requirements from multiple stakeholders. More specifically, they propose a tool, named CARL, that automatically translates NL into logic and then uses theorem proving and model checking to detect inconsistency in the requirements. Popescu et al. [61] developed a semiautomated process for reducing ambiguity in software requirements using object-oriented modeling. They present a three-step approach to reducing the number of ambiguities, inconsistencies and underspecifications. They first parse NL requirements using a constraining grammar. Then, they elicit relevant terms to create an object-oriented model, which is used to detect ambiguities and inconsistencies.

7.2.2 Unconstrained NL

The previously cited works mainly focus on constrained NL, or on formal and semiformal languages. Let us now look at some of the works that focus on preventing and detecting ambiguities in unconstrained NL. Some of the studies have a specific focus, i.e., they address how to handle some of those mistakes that are typically in the analysts' list when they are reviewing a requirements documents. In particular, Berry and Kamsties [9] explain how to handle the syntactically dangerous "all" and plural in requirements, Neumann [54] covers the dangerously misplaced "only," and Rupp and Goetz [66] discuss the semantically dangerous "all."

Rule-based approaches Other works are based on the definition of a set of language *rules* that allow to identify typically ambiguous words and constructions. Stemming from the work of Berry et al. [10], Gnesi et al. [33] developed QuARS, a tool that detects ambiguity and vagueness based on keyword-based linguistic indicators. The tool mainly

focuses on vague terms and on potential sources of referential ambiguities and coordination ambiguities, which are particular types of syntactic ambiguities involving the improper usage of coordinating conjunctions (e.g., "and," "or"). Kiyavitskaya et al. [44] propose a two-step approach to identifying ambiguities in NL requirements. In the first step, a tool applies a set of ambiguity measures to the requirements, in order to identify potentially ambiguous sentences. In the second step, a (manually simulated) tool shows the specific parts that are potentially ambiguous in the set of sentences identified. As in the work of Gnesi et al. [33], the approach leaves the final decision of ambiguity to human users, but provides them with further elements to decide. Analogously, Tjong and Berry [76] propose a tool, SREE (Synthesized Requirements Engineering Environment), that detects potential ambiguities in the requirements document and report them to the user, who has to correct the requirements if he/she believes that they are truly ambiguous. Following the same idea of Kiyavitskaya et al. [44], Gleich et al. [31] propose a Web-based lightweight tool to detect ambiguities and to explain the ambiguity sources. When detecting ambiguities, the tool basically relies on a grep-like technique, which makes it highly reliable, applicable to different languages, and independent from error-prone natural language parsing. For every detected ambiguity, the tool provides an explanation of the reason why the detected item represents a potential problem. Following the rationale of these research tools, Sirius Requirements developed Requirements Assistant^{TM}[72], which, to our knowledge, is the only commercial tool that supports NL requirements analysis. Finally, it is worth mentioning the recent work of Arora et al. [5], which presents 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.

Statistical approaches Another group of works uses *statistical* techniques to identify ambiguities. Chantree et al. [12] present a technique that helps requirements analysts to identify nocuous ambiguities by automatically identifying innocuous ambiguities, and reporting only the remaining nocuous ones. The focus of this work is on *coordination* ambiguities, and a set of heuristics, developed according to a dataset built by human assessors, is presented to discriminate between nocuous and innocuous ambiguities. This approach was extended for *referential* ambiguities—referred as *anaphoric* ambiguities—-by Yang et al. [78, 79]. The authors propose a number of heuristics, each of which captures information that may lead a reader to favor a particular interpretation of the text and use them to build a classifier. The classifier predicts the degree to which particular interpretations are preferred. Ferrari et al. [21, 22] propose to detect pragmatic ambiguities in NL requirements defined for a specific application domain. The idea is to first perform a Web search to retrieve a set of documents focused on the same domain of the requirements. These documents are used to build multiple knowledge graphs, which model the domain knowledge of potential readers of the requirements. Then, each requirement is enriched with the domain knowledge of the different graphs. Finally, these enriched representations of a requirement are compared to detect potential ambiguities.

Observations The main difference between rule-based approaches and statistical approaches resides on the fact that the former find *all* the potential ambiguities, within the limits of the rules, and let the analyst decide whether the detected ambiguity is nocuous or not. Instead, the latter are oriented to automatically discard innocuous ambiguities. In other terms, rule-based approaches are designed to bring 100 % recall, while statistical approaches aim at maximizing precision. The usage of statistical techniques in requirements analysis has been questioned by several authors (see, e.g., [8, 67]). In particular, for ambiguity detection, Berry et al. [8] advocate the usage of clerical, rule-based tools to detect ambiguities that can be identified without producing false negatives (i.e., without leaving ambiguities undiscovered). If part of the ambiguity detection task cannot be performed without false negatives, that part shall be left to the thinking analyst. Though we agree with Berry's arguments, we observe that certain types of ambiguities, and, in particular, context-dependent ones (i.e., pragmatic), have a inherent statistical nature—since contexts are uncountable—and, except for a limited set of cases, are hardly detectable with infallible rule-based tools. Hence, for these types of ambiguities, we should rely on fallible non-rule-based tools, which shall however be tuned to maximize recall.

In this review, we have mentioned only part of the works on ambiguity prevention and detection. A complete overview of the currently available techniques can be found in a recent survey [71]. At its current stage, our work does not propose a solution to prevent or detect ambiguity. On the other hand, our work, and our examples, can be used as a starting point for further research on ambiguity detection tools, which are mainly focused on detecting vagueness, syntactic and referential ambiguities, and do not account for many of the cases that we encountered in interviews.

7.3 Requirements elicitation interviews

Theoretical (e.g., [16]) and empirical (e.g., [1, 11]) studies have been performed to understand the relevance of adequate communication during requirements elicitation and

to assess the effectiveness of interviews in eliciting requirements. Davis et al. [17] perform a systematic literature review on previous empirical works concerning requirements elicitation techniques and conclude that structured interviews appear as the most effective strategy for elicitation. Moreover, Portugal [62] provides a large set of guidelines, based on the author's experience, to conduct a successful interview.

Contextual factors Other works focus on contextual factors and communication defects that might affect knowledge transfer. Among them, Coughlan and Macredie [16] identifies *articulation* (i.e., difficulty in expressing concepts) *misunderstanding* (i.e., divided interpretation on the same piece of information) and *conflict* (i.e., disagreements due to different viewpoints) as the general classes of problems that hamper communication during requirements elicitation. Distanont et al. [18] makes a more fine-grained analysis of the contextual factors affecting knowledge transfer and identifies three main classes of factors, namely *human-oriented* (e.g., trust between customer and analyst, expressive ability of the customer, absorptive capacity of the analyst), *process-oriented* (e.g., nature of information to be transferred, time constraints) and *context-oriented* (e.g., culture diversity, executive support commitment). In general, these works have a rather broad perspective and treat the different factors in conjunction, without specifically investigating one particular class of factors.

Domain knowledge A more focused approach appears in the work of Hadar et al. [36], in which the authors analyze the effect of common domain knowledge between analyst and stakeholders in interviews. The authors' observation is that, even though such knowledge is commonly assumed to have positive effects on the requirements engineering process, it can cause some negative effects as well. On the one hand, while being familiar with the domain, the analyst prepares more focused questions for the interview and uses a domain-specific language, which is shared with the customer. Moreover, domain knowledge can help in directing the interview, since it gives to the interviewer the ability to promptly assess the information received by the stakeholders, in terms of both relevance and clarity. On the other hand, domain knowledge might lead the analyst toward a fixed point of view, in the sense that the analyst is biased to think that he/she knows the answers better than the customer, neglecting to incorporate relevant information. Moreover, given his/her experience in the domain, the analyst might perceive some questions as trivial. Avoiding to ask trivial questions can lead to implicit assumptions and incomplete requirements. The findings of Hadar et al. [36] are in line with the empirical observations of Berry et al. [55, 56], in which the authors validate the intuition that the presence of a *smart ignoramus*—i.e., a domain ignorant with critical sense—in a requirements engineering team helps in spotting out inconsistencies, and contributes to the success of a project.

Observations Overall, our work mainly differs from the literature in interviews for its focus, namely ambiguity. Indeed, though Distanont et al. [18] mentions ambiguity among the factors affecting knowledge transfer, we are not aware of any study that specifically investigates ambiguity in interviews. Our work contributes to the research on interview methods by highlighting the positive role of ambiguity during requirements elicitation. Moreover, in relation to the work of Hadar et al. [36], we reveal the part played by the domain knowledge dimension in the perception of ambiguity from the analyst's point of view. In this sense, our quantitative data in Sect. 4.5 suggest that domain knowledge plays a role on the types of ambiguity that occur in interviews. Empirical validation of this early finding might clarify the relation between ambiguity, degree of domain knowledge of the analyst and interview success.

7.4 Tacit knowledge

The notion of tacit knowledge was first introduced by Polany [60]. The essay speaks about the role of language in communicating knowledge and tells that some type of knowledge has limited capability for transfer, and, hence, it has a tacit component. He makes the example of art, which can be transferred only by example, from expert to apprentice. In this sense, tacit knowledge can be regarded as *know-how* [46]. Nevertheless, as noted by the analysis of Grant [34], for Polany each knowledge has a tacit part, and, in his vision, knowledge has multiple shades of tacitness, which go from *ineffable* knowledge—i.e., when articulation is not possible—to explicit knowledge. However, to make the concept of tacit knowledge applicable in practical domains, a sort of dichotomy between tacit and explicit knowledge has been introduced, as, e.g., in corporate settings [57]. Though some differences exist in the interpretations of these concepts, explicit knowledge is normally referred as the knowledge that can be aggregated and written down, while tacit knowledge is the knowledge that has some difficulty in being written down, and hence, transferred [28]. This difficulty may change depending on the situation; however, extracting tacit knowledge is generally recognized as a bottleneck since the start of a project, as shown for expert systems [40] and knowledge management systems [63].

Tacit knowledge and RE Since one of the goals of a RE process is making explicit what is implicit—or considered as understood by the stakeholders (i.e., domain knowledge,

goals, requirements)—the concept of tacit knowledge is a matter of interest for the RE community, and approaches have been developed to disclose tacit knowledge (see, e.g., Rugg et al. [65], and Grunbacher and Briggs [35]). Along this line of research, Gervasi et al. [28] proposed a framework based on predicates to reason on the notion of tacit knowledge in RE. The framework defines a set of predicates to model the communication between a customer and a requirements analyst. Based on the truth value of these predicates, the framework distinguishes multiple practical cases in which tacit knowledge surfaces, and suggests strategies to elicit such knowledge. As mentioned, our work stems from this framework and focuses on one aspect, i.e., ambiguity, that is only touched upon in the work of Gervasi et al. [28]. In this sense, our paper can be seen as an independent follow-up of the mentioned work. Among the other papers that followed Gervasi et al. [28], it is worth citing the work of Sutcliffe and Sawyer [75], which surveys the different requirements elicitation techniques available, in light of the tacit knowledge framework [28]. Sutcliffe and Sawyer [75] suggest the following research directions to address the challenge of tacit knowledge and unknown unknowns. For *green-field* applications, in which unknown unknowns might be dominant, they encourage the development of creativity techniques and their integration with socio-technical models. For *brown-field* applications, in which tacit knowledge appears to be the main problem, text mining and IR techniques are encouraged. Indeed, the idea is that in brown-fields, the required knowledge is documented somewhere, but is not easily accessible. Overall, they recommend the usage of rich-media and interactive elicitation approaches to extend the common ground between the different stakeholders. In light of our experience in interviews, we particularly agree on the need for rich-media approaches. Indeed, in our interviews we did not use graphical tools, but the analyst often felt that a simple informal diagram would have helped in spotting out missing information, in particular about the domain. This need can be also explained in light of our framework. In our domain component, we included a business rules sub-component (in D-Rules, Fig. 1), but we did not include a *business process* sub-component. Indeed, the actual processes of the domain appear hard to be mentally visualized in their entirety, unless one does not unfold them explicitly with the support of a diagram. In other terms, our framework takes into account the intuition that the capability of our brains is limited, especially when we have to consciously analyze a complex process. Hence, external tools become key assets. This need for external means of representations applies also to system requirements. This observation is in line with the solution proposed by Zhang et al. [80], another work that follows the footsteps of Gervasi et al. [28] and of Sutcliffe and Sawyer [75], and that presents a meta-model approach for identifying missing information in requirements. The work highlights that not only external tools—such as diagrams, or models in general—are required to disclose tacit knowledge, but also meta-models are needed to reason on the information that requirements models might have left behind.

Our contribution to the research on tacit knowledge in RE resides on the identification of ambiguity as a tool for disclosing tacit knowledge. Of course, language ambiguity is *one* of the tools, and, as pointed out by Sutcliffe and Sawyer [75], the challenge of *unknowns* is still open.

8 Conclusion

This paper presents a phenomenology of ambiguity in requirements elicitation interviews and stresses the primary role of ambiguity in disclosing tacit knowledge. Our vision is based on the conviction, in line with the arguments of Chantree et al. [12] and of Massey et al. [49], that ambiguity is a subjective phenomenon, which derives from the relation between the signifier (i.e., speech fragment, word or sentence) and the reader (or listener), who assigns a meaning to the signifier. Moreover, our view shows that ambiguity is also a contextual and situational phenomenon, in which the pragmatic facet of the recipient of the signifier plays a primary role. At this stage of the research, requirements analyst shall be aware that what they normally call ambiguity can have different names, namely interpretation unclarity, acceptance unclarity, multiple understanding, detected and undetected incorrect disambiguation, and correct disambiguation. Naming concepts is a primary means to understand relevant aspects of our reality. Hence, given the relevance of ambiguity in requirements engineering, we argue that these different names, and the meanings that they convey, can be useful to foster a deeper understanding of the ambiguity phenomenon. As the research matures, we expect to give requirements analysts a clearer view of their perception of the customer's words, and of the potential benefits of any misunderstanding. Our emphasis on the role of ambiguity in disclosing tacit knowledge is not merely speculative. Indeed, humans tend to overlook ambiguity and subconsciously take an effort to assign a meaning to what they perceive [36]. Our near future commitment is to provide conceptual models to further clarify the phenomenon. Such models will enable analysts to achieve a conscious skeptical attitude [43, 68], to go beyond the surface meaning of what they hear, and ultimately access the unknown.

Acknowledgments The authors would like to thank Daniel M. Berry for his precious recommendations and all the anonymous reviewers who helped improving this paper. This work was partially supported by the LearnPAd FP7-ICT-2013.8.2 European Project.

References

1. Agarwal R, Tanniru MR (1990) Knowledge acquisition using structured interviewing: an empirical investigation. JMIS 7(1):123–140
2. Alves CF, Pereira S, Valença G, Pimentel J, de Andrade RV (2007) Preliminary results from an empirical study in market-driven software companies. In: WER'07, pp 127–134
3. Ambriola V, Gervasi V (2006) On the systematic analysis of natural language requirements with Circe. ASE 13
4. Aristotle (1984) On sophistical refutations. In: Barnes J (ed) The complete works of Aristotle: the revised Oxford translation. Princeton University Press, Princeton, New Jersey
5. Arora C, Sabetzadeh M, Briand L, Zimmer F (2015) Automated checking of conformance to requirements templates using natural language processing. IEEE Trans Softw Eng 41(10):944–968
6. Berry D (2008) Ambiguity in natural language requirements documents. In: Paech B, Martell C (eds) Innovations for requirement analysis. From stakeholders needs to formal designs, LNCS, vol 5320. Springer, Berlin, pp 1–7
7. Berry D, Kamsties E (2004) Ambiguity in requirements specification. In: Sampaio do Prado Leite JC, Doorn JH (eds) Perspectives on software requirements. The Springer International Series in engineering and computer science, vol 753. Springer, New York, pp 7–44
8. Berry DM, Gacitua R, Sawyer P, Tjong SF (2012) The case for dumb requirements engineering tools. In: Regnell B, Damian D (eds) REFSQ, LNCS, vol 7195. Springer, pp 211–217
9. Berry DM, Kamsties E (2005) The syntactically dangerous all and plural in specifications. IEEE Softw 22(1):55–57
10. Berry DM, Kamsties E, Krieger MM (2003) From contract drafting to software specification: linguistic sources of ambiguity
11. Browne GJ, Rogich MB (2001) An empirical investigation of user requirements elicitation: comparing the effectiveness of prompting techniques. JMIS 17(4):223–249
12. Chantree F, Nuseibeh B, Roeck AND, Willis A (2006) Identifying nocuous ambiguities in natural language requirements. In: RE'06, pp 56–65
13. Cimatti A, Roveri M, Susi A, Tonetta S (2011) Formalizing requirements with object models and temporal constraints. SoSyM 10(2):147–160
14. Clark HH (1996) Using language. Cambridge University Press, Cambridge
15. Corvera Charaf M, Rosenkranz C, Holten R (2013) The emergence of shared understanding: applying functional pragmatics to study the requirements development process. ISJ 23(2):115–135
16. Coughlan J, Macredie RD (2002) Effective communication in requirements elicitation: a comparison of methodologies. Requir Eng 7(2):47–60
17. Davis A, Dieste O, Hickey A, Juristo N, Moreno AM (2006) Effectiveness of requirements elicitation techniques: empirical results derived from a systematic review. In: RE'06. IEEE, pp 179–188
18. Distanont A, Haapasalo H, Vaananen M, Lehto J (2012) The engagement between knowledge transfer and requirements engineering. IJKL 1(2):131–156
19. Empson W (1966) Seven types of ambiguity. New Directions Paperbook, New York
20. Ferrari A, dell'Orletta F, Spagnolo GO, Gnesi S (2014) Measuring and improving the completeness of natural language requirements. In: REFSQ'14, LNCS, vol 8396. Springer, pp 23–38
21. Ferrari A, Gnesi S (2012) Using collective intelligence to detect pragmatic ambiguities. In: RE'12. IEEE, pp 191–200
22. Ferrari A, Lipari G, Gnesi S, Spagnolo GO (2014) Pragmatic ambiguity detection in natural language requirements. In: AIRE'14. IEEE, pp 1–8
23. Ferrari A, Spoletini P, Gnesi S (2015) Ambiguity as a resource to disclose tacit knowledge. In: 2015 23rd IEEE international requirements engineering conference (RE). IEEE, pp 26–35
24. Friedrich WR, Van Der Poll JA (2007) Towards a methodology to elicit tacit domain knowledge from users. IJIKM 2(1):179–193
25. Gacitua R, Ma L, Nuseibeh B, Piwek P, De Roeck A, Rouncefield M, Sawyer P, Willis A, Yang H (2009) Making tacit requirements explicit. In: MARK'09. IEEE, pp 40–44
26. Gacitua R, Sawyer P, Gervasi V (2011) Relevance-based abstraction identification: technique and evaluation. Requir Eng 16(3):251–265. doi:10.1007/s00766-011-0122-3
27. Gause D, Weinberg G (1989) Exploring requirements: quality before design. Dorset House Pub
28. Gervasi V, Gacitua R, Rouncefield M, Sawyer P, Kof L, Ma L, Piwek P, De Roeck A, Willis A, Yang H et al (2013) Unpacking tacit knowledge for requirements engineering. In: Maalej W, Thurimella AK (eds) Managing requirements knowledge. Springer, pp 23–47
29. Gervasi V, Zowghi D (2005) Reasoning about inconsistencies in natural language requirements. ACM Trans Softw Eng Methodol 14(3):277–330
30. Gleich B, Creighton O, Kof L (2010) Ambiguity detection: towards a tool explaining ambiguity sources. In: REFSQ'10, LNCS, vol 6182. Springer, pp 218–232
31. Gleich B, Creighton O, Kof L (2010) Ambiguity detection: towards a tool explaining ambiguity sources. In: Requirements engineering: foundation for software quality. Lecture notes in computer science, vol 6182. Springer, Berlin, pp 218–232. http://dx.doi.org/10.1007/978-3-642-14192-8_20
32. Glinz M, Fricker SA (2014) On shared understanding in software engineering: an essay. CSRD, pp 1–14
33. Gnesi S, Lami G, Trentanni G (2005) An automatic tool for the analysis of natural language requirements. IJCSSE 20(1)
34. Grant KA (2007) Tacit knowledge revisited—we can still learn from Polanyi. Electron J Knowl Manag 5(2):173–180
35. Grunbacher P, Briggs RO (2001) Surfacing tacit knowledge in requirements negotiation: experiences using easywinwin. In: Proceedings of the 34th annual Hawaii international conference on system sciences, 2001. IEEE, 8pp
36. Hadar I, Soffer P, Kenzi K (2014) The role of domain knowledge in requirements elicitation via interviews: an exploratory study. Requir Eng 19(2):143–159
37. Harwell R, Aslaksen E, Mengot R, Hooks I, Ptack K (1993) What is a requirement? INCOSE Int Symp 3(1):17–24
38. Hay D, Healy KA, Hall J et al (2000) Defining business rules—What are they really? Technical report Rev 1.3, the Business Rules Group
39. Hickey AM, Davis AM (2004) A unified model of requirements elicitation. J Manag Inf Syst 20(4):65–84
40. Horvath JA (2000) Working with tacit knowledge. Knowl Manag Yearb 2000–2001:34–51
41. Ide N, Véronis J (1998) Introduction to the special issue on word sense disambiguation: the state of the art. Comput Linguist 24(1):2–40
42. Kamsties E (2005) Understanding ambiguity in requirements engineering. In: Engineering and managing software requirements. Springer, Berlin, pp 245–266
43. Karten N (2013) Managing expectations: working with people who want more, better, faster, sooner, Now! Addison-Wesley

44. Kiyavitskaya N, Zeni N, Mich L, Berry DM (2007) Requirements for tools for ambiguity identification and measurement in natural language requirements specifications. In: WER'07, pp 197–206
45. Kof L (2010) From requirements documents to system models: a tool for interactive semi-automatic translation. In: RE'10
46. Kogut B, Zander U (1992) Knowledge of the firm, combinative capabilities, and the replication of technology. Org Sci 3(3):383–397
47. van Lamsweerde L (2009) Requirements engineering—from system goals to UML models to software specifications. Wiley, London
48. Maiden N, Rugg G (1996) ACRE: selecting methods for requirements acquisition. Softw Eng J 11(3):183–192
49. Massey AK, Rutledge RL, Anton AI, Swire PP (2014) Identifying and classifying ambiguity for regulatory requirements. In: RE'14. IEEE, pp 83–92
50. Mich L (1996) NL-OOPS: from natural language to object oriented requirements using the natural language processing system LOLITA. NLE 2(2):161–187
51. Mich L, Franch M, Inverardi PN (2004) Market research for requirements analysis using linguistic tools. Requir Eng 9(1):40–56
52. Mich L, Garigliano R (2000) Ambiguity measures in requirements engineering. In: ICS'00, 16th IFIP WCC, pp 39–48
53. Mühlhäusler P (1986) Pidgin and creole linguistics. Blackwell, Oxford
54. Neumann PG (1986) Only his only grammarian can only say only what only he only means. ACM SIGSOFT SE Notes 9(1):6
55. Niknafs A, Berry DM (2012) The impact of domain knowledge on the effectiveness of requirements idea generation during requirements elicitation. In: 2012 20th IEEE international requirements engineering conference (RE). IEEE, pp 181–190
56. Niknafs A, Berry DM (2013) An industrial case study of the impact of domain ignorance on the effectiveness of requirements idea generation during requirements elicitation. In: 2013 21st IEEE international requirements engineering conference (RE). IEEE, pp 279–283
57. Nonaka I (1991) The knowledge-creating company. Harvard Bus Rev 69(6):96–104
58. Osborne M, MacNish CK (1996) Processing natural language software requirement specifications. pp 229–236
59. Pitts MG, Browne GJ (2004) Stopping behavior of systems analysts during information requirements elicitation. J Manag Inf Syst 21(1):203–226
60. Polanyi M (1966) The tacit dimension. Doubleday, Garden City
61. Popescu D, Rugaber S, Medvidovic N, Berry D (2008) Reducing ambiguities in requirements specifications via automatically created object-oriented models. In: Innovations for requirement analysis. From stakeholders needs to formal designs, lecture notes in computer science, vol 5320. Springer, Berlin, pp 103–124
62. Portugal S (2013) Interviewing users: how to uncover compelling details. Louis Rosenfeld
63. Riege A (2005) Three-dozen knowledge-sharing barriers managers must consider. J Knowl Manag 9(3):18–35
64. Robertson S, Robertson J (2012) Mastering the requirements process: getting requirements right. Addison-Wesley, Boston
65. Rugg G, McGeorge P, Maiden N (2000) Method fragments. Expert Syst 17(5):248–257
66. Rupp C, Goetz R (2000) Linguistic methods of requirements-engineering (nlp). In: Proceedings of European software process improvement conference (EuroSPI)
67. Ryan K (1993) The role of natural language in requirements engineering. In: Proceedings of IEEE international symposium on requirements engineering, 1993, pp 240–242
68. Saiedian H, Dale R (2000) Requirements engineering: making the connection between the software developer and customer. Inf Softw Technol 42(6):419–428
69. Schneider GM, Martin J, Tsai WT (1992) An experimental study of fault detection in user requirements documents. ACM Trans Softw Eng Methodol 1(2):188–204
70. Sennet A (2015) Ambiguity. In: Zalta EN (ed) The Stanford encyclopedia of philosophy, spring 2015 edition
71. Shah US, Jinwala DC (2015) Resolving ambiguities in natural language software requirements: a comprehensive survey. SIGSOFT Softw Eng Notes 40(5):1–7
72. Sirius Requirements: http://www.sirius-requirements.com
73. Software Engineering Technology Committee and Institute of Electrical and Electronics Engineers (1994) IEEE recommended practice for software requirements specifications. IEEE Std 830-1998. Institute of Electrical and Electronics Engineers. IEEE Computer Society
74. Sommerville I, Sawyer P (1997) Viewpoints: principles, problems and a practical approach to requirements engineering. Ann Softw Eng 3(1):101–130
75. Sutcliffe A, Sawyer P (2013) Requirements elicitation: towards the unknown unknowns. In: RE'13. IEEE, pp 92–104
76. Tjong S, Berry D (2013) The design of SREE a prototype potential ambiguity finder for requirements specifications and lessons learned. In: Doerr J, Opdahl A (eds) Requirements engineering: foundation for software quality, lecture notes in computer science, vol 7830. Springer, Berlin, pp 80–95
77. Wilson WM, Rosenberg LH, Hyatt LE (1997) Automated analysis of requirement specifications. In: ICSE'97, pp 161–171
78. Yang H, De Roeck A, Gervasi V, Willis A, Nuseibeh B (2010) Extending nocuous ambiguity analysis for anaphora in natural language requirements. In: RE'10. IEEE, pp 25–34
79. Yang H, Roeck AND, Gervasi V, Willis A, Nuseibeh B (2011) Analysing anaphoric ambiguity in natural language requirements. Requir Eng 16(3):163–189
80. Zhang Z, Thanisch P, Nummenmaa J, Ma J (2014) Detecting missing requirements in conceptual models. In: Dregvaite G, Damasevicius R (eds) Information and software technologies. Springer, Berlin, pp 248–259
81. Zowghi D, Coulin C (2005) Requirements elicitation: a survey of techniques, approaches, and tools. In: Engineering and managing software requirements. Springer, Berlin, pp 19–46
82. Zowghi D, Gervasi V, McRae A (2001) Using default reasoning to discover inconsistencies in natural language requirements. In: APSEC 2001 eighth Asia-Pacific software engineering conference, 2001, pp 133–140