Ambiguity as a Resource to Disclose Tacit Knowledge

Alessio Ferrari
ISTI-CNR
Pisa, Italy
alessio.ferrari@isti.cnr.it

Paola Spoletini
Kennesaw State University
Georgia, USA
pspoleti@kennesaw.edu

Stefania Gnesi
ISTI-CNR
Pisa, Italy
stefania.gnesi@isti.cnr.it

***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 analyse the role of ambiguity in requirements elicitation interviews. To this end, we have performed a set of customer-analyst interviews to observe how ambiguity occurs during requirements elicitation. From this direct experience, we have observed that ambiguity is a multi-dimensional cognitive phenomenon with a dominant *pragmatic* facet, and we have defined a phenomenological framework to describe the different types of ambiguity in interviews. We have also discovered that, rather than an obstacle, the occurrence of an ambiguity is often a *resource* for discovering tacit knowledge. Starting from this observation, we have envisioned the further steps needed in the research to exploit these findings.**

I. 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 [1], [2]. Among the available requirements elicitation techniques (e.g., workshops, focus groups, scenarios, prototypes [3], [4]), interviews with stakeholders are the most commonly used [5]–[8], and are considered among the most effective for knowledge transfer [9]–[12]. Normally, requirements elicitation interviews involve two roles: a customer and a requirements analyst. Several factors have been observed to negatively affect the interview process, from the trustworthiness and motivation of the customer, to the absorptive capacity of the requirements analyst [2]. Among such factors, ambiguity in communication is regarded as a major obstacle [2] 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 [13].

Past works on ambiguity in requirements engineering have been mainly focused on natural language (NL) ambiguities in requirements documents (i.e., textual documents) [14]–[31]. Part of these works are focused on the identification of typical ambiguous terms and constructions [15], [16], [19], [20], [24]. Other works address the ambiguities by translating the requirements into formal languages or models [22], [23], [26]. Finally, some works focus on the usage of NL understanding methodologies [17], [18] and on artificial intelligence techniques [27]–[30]. 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 have seen that the concept of ambiguity in NL requirements documents, and its classical lexical, syntactic, semantic clues [16], were accounting for a very limited set of ambiguity phenomena that occur at the level of requirements elicitation, where 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.* [32] on *tacit knowledge*. Tacit knowledge in requirements engineering [2], [9], [32] is defined as the knowledge that a customer has, but does not pass to the requirements analyst. Tacit knowledge is regarded as a major problem in requirements elicitation, and, though process solutions exist [9], means are required to *improve* the detection of tacit knowledge. In our study, we have 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 conversation picklock to disclose 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 [33].

The paper is organised as follows. Sect. II informally defines ambiguity in requirements elicitation interviews, and presents the contextual aspects that are useful to understand our vision of ambiguity. In Sect. III, we more formally define ambiguity by instantiating the framework for tacit knowledge defined in

[32] in the context of customer-analyst interviews. Sect. IV describes the different categories of ambiguities in interviews. Sect. V explains the role of ambiguity in disclosing tacit knowledge. Sect. VI presents the research challenges that this work opens, and Sect. VII concludes the paper.

II. 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 have defined for ambiguity.

A. Ambiguity in Interviews

Requirements elicitation interviews normally involve a customer and a requirements analyst, and the elicitation process consists of a dialogue where 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 [4]. In this work, we focus on unstructured interviews, where 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 by different subjects. In interviews, ambiguities are associated to *misunderstanding* situations, when an expression of a customer is either not understood or incorrectly interpreted by the requirements analyst. Let us give an informal definition of ambiguity in requirements elicitation interviews.

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 such 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. This definition includes also those cases in which the analyst *cannot* assign a meaning to the unit of information expressed by the customer.

Notice that the above definition takes into account only the ambiguity cases associated to expressions of the customer that are misunderstood by the analyst, and does not consider the situations where 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 previous domain knowledge [5], this simplification avoids us to consider aspects associated to dialogue and argumentation – addressed in [34] – which would be beyond our scope.

B. Interviews

The definition given above has been used as a reference to perform our inquiry concerning ambiguity in requirements elicitation interviews. To study the problem, we simulated 27 unstructured interviews, where the customers were asked to come to the meeting with one or more ideas of software-intensive systems to develop. The interviews have been always performed by the same requirements analyst (i.e., the first author), whose previous research focused on the detection of ambiguity in textual requirements [29], [30], to have a uniform perception of the ambiguity cases. The role of the customer was played by 4 domain-experts (in History of Arts, Public Administration, Health-care and Training Courses) and 7 software engineers. The domain experts were asked to provide ideas in their domains, while the software engineers could bring novel ideas in domains in which they felt familiar. 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 idea and, when domain-related aspects emerged that appeared new to the analyst, this latter 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 have isolated the speech fragments that were perceived as ambiguous by the analyst (132 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 categorisation of ambiguities in interviews, by iteratively refining the model of tacit knowledge in [32] on the basis of the observed ambiguities.

The interviews were not meant to be a rigorous empirical study, but were used as inspiration to categorise ambiguities. This is a first step towards a broader research, as envisioned in the challenges described in Sect. VI. 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 confute it, taking into account the story that generated it.

C. The Pragmatic Facet

In our on-field observation during the performed 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" [16], [19]), some were due to the usage of universal quantifiers (e.g., "all" and plurals [15]), and few were anaphoric ambiguities [28]. 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. In other terms, they were *pragmatic ambiguities* [29], [30].

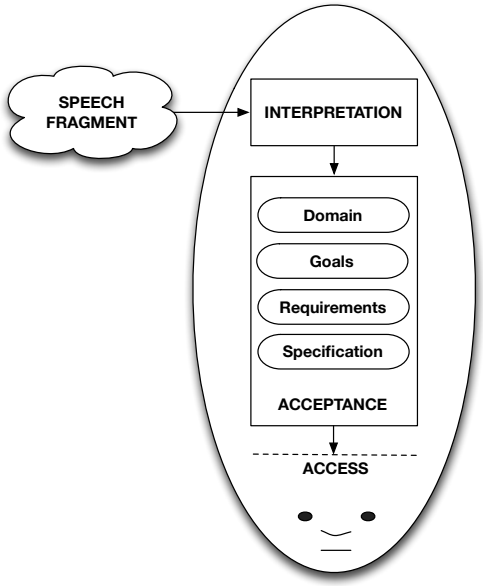

Fig. 1: A model of the process of access to a unit of information by a requirements analyst.

To understand these situations, it is useful to refer to Figure 1, where 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. II-A – is perceived by the analyst. The access to the information (Access line in Figure 1) implies that the analyst first gives an interpretation (Interpretation block) and then considers if such interpretation is acceptable in light of his current mental framework (Acceptance block). Indeed, as highlighted in [35], and as we have directly 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. Such mental framework includes the other requirements currently expressed by the customer (Requirements), the motivations of such requirements (Goals, in goal-oriented requirements engineering terms [36]), 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.

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 if 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 such interpretation might be different from the intended meaning of the customer. If such 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.

To account for the different types of situations that we have seen in practice, which depend on the interaction of the different blocks depicted in Figure 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 in [32], in the context of customer-analyst interviews, and we extend it to account for the different cases of ambiguities that we have encountered in practice. In the following, we will give a precise definition of ambiguity, and will provide a categorisation of the phenomenon.

III. Definition of Ambiguity

The framework for tacit knowledge defined by Gervasi *et al.* [32] introduces a set of predicates that can be associated to a unit of information k. Such 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" [32]. 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 to the system to develop. Now, 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[1]. Consider that k can belong either to the requirements for the system or to the domain of the system. Moreover, consider i as a speech fragment.

From the predicates associated to k we select two predicates useful for our definition of ambiguity:

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

As in [32], $accessible_{a,i}(k)$ 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.

A. 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 give a meaning to i) and *acceptable* in the current mental framework of the analyst. More formally:

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

[1]In [32], i refers to the whole *interview*. Here, i is associated to the specific piece of the interview (i.e., the speech fragment) where k is articulated.

Note that $acceptable_{a,i}(k) \implies interpretable_{a,i}(k)$, since the information can be accepted only after it has been interpreted.

B. Interpretable

The expression of k is interpretable by the analyst (i.e., $interpretable_{a,i}(k)$) if the articulation of the information can be parsed by the analyst, i.e., 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 two examples.

Example III.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 realise a system for associating the paintings to the authors, to support the process of attribution. He tells that, to perform attribution of a painting to an artist, he applies the "*connoisseurship* method" (a method based on using previous paintings to analyse 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 to the "*connoisseurship* method".

Example III.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.

C. Acceptable

The fact that the analyst can give an interpretation to the utterance (i.e., speech fragment) of the customer does not imply that such utterance is acceptable in his 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 to 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 where we have $\neg acceptable_{a,i}(k)$.

Example III.3 ($\neg acceptable^{G}_{a,i}(k)$)**.** These are the cases where 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 left for 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 if the waiting time was due to 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 III.4 ($\neg acceptable^{R}_{a,i}(k)$)**.** 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 told: *"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 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. IV-C), was discovered thanks to another ambiguity (i.e., an *acceptance unclarity*, see Sect. IV-A).

Example III.5 ($\neg acceptable^{D}_{a,i}(k)$)**.** These are situations where the speech fragment of the customer is inconsistent with the domain knowledge of the analyst. In our examples, 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 told: *"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.

Example III.6 ($\neg acceptable^{S}_{a,i}(k)$)**.** 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 swim-

ming training. He told: *"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 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"*.

D. 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 formal definition of ambiguity that accounts for all the cases perceived in our interviews.

Let the notation 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:

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

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 to such 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 precise definition mimics the informal definition of ambiguity given in Section II-A.

From this definition, we can derive six different feasible main cases. 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:

$$\neg interpretable_{a,i}(k) \vee \neg acceptable_{a,i}(k) \vee interpretable_{a,i}(k')$$

In practice, when either the speech fragment is not interpretable, or not acceptable, or when it can be interpreted in a different way with respect to the intended meaning of the customer. Such cases are summarized in Table I. The table considers only the feasible cases, since we discard the combination in which we have $acceptable_{a,i}(k) \wedge \neg interpretable_{a,i}(k)$.

IV. Categories of Ambiguities

This section discusses the categories of ambiguities derived from the formal definition introduced in the previous section. We have classified them into three main classes, namely *unclarity*, *multiple understanding*, and *incorrect disambiguation*, and in the remainder of this section we present them accordingly.

A. Unclarity

The "unclarity" class includes those situations where the requirements analyst cannot give any interpretation or acceptable meaning to the unit of information expressed, either because the information has not been articulated in clear language, or for the usage of domain jargon, or because the interpretation is not acceptable in his/her mental framework. This type of ambiguity can be formally represented as:

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

If the utterance cannot be interpreted because it includes domain jargon that is unknown to the analyst, or vague expressions, we have $\neg interpretable_{a,i}(k)$, which causes the unclarity. In these situations, exemplified in Ex. III.1 and Ex. III.2, 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 have been presented from Ex. III.3 to Ex. III.6. In all the cases were we have $\neg acceptable_{a,i}(k)$, we speak about *acceptance unclarity*.

B. Multiple Understanding

The class "multiple understanding" includes all the cases in which the 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 have been experienced in our interviews. Here, we give an example.

Example IV.1 (multiple understanding)**.** One of our customers wanted to define a Web-based platform where the citizens can send suggestions for laws to the parliament. The customer told 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"*.

In our experience, we have noticed that in some cases the customer articulated an idea that had multiple meanings for the analyst, and each one resulted valid. Though not evident from the formalization, such situation is accounted in our definition of multiple understanding. An example might help clarifying these situations.

Example IV.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); (2) the application should be

TABLE I: Summary of ambiguity phenomena

k		k'		**Phenomenon**
$\neg interpretable(k)$	$\neg acceptable(k)$	$\neg interpretable(k')$	-	interpretation unclarity
$interpretable(k)$	$\neg acceptable(k)$	$\neg interpretable(k')$	-	acceptance unclarity
$interpretable(k)$	$acceptable(k)$	$interpretable(k')$	$acceptable(k')$	multiple understanding
			$\neg acceptable(k')$	detected incorrect disambiguation
-	$\neg acceptable(k)$	$interpretable(k')$	$acceptable(k')$	undetected incorrect disambiguation
			$\neg acceptable(k')$	detected incorrect disambiguation

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 told "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 where the two ideas could be exclusive ($k' = k_1 \veebar k_2$, where $\veebar$ is the logical XOR operator). Therefore, also this situation falls in the category of multiple understanding.

C. Incorrect Disambiguation

The situations in which the analyst assigns a *single* interpretation to the expression of the customer, but such interpretation is different from the meaning intended by the customer are classified as "incorrect disambiguation" situations. 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 case of ambiguity normally includes *unconscious disambiguation* phenomena [16], which are hard to identify, unless the requirements analyst suspects that his/her interpretation of the expression of the customer is not correct. Such phenomenon occurs when (1) the analyst can give an interpretation to the utterance of the customer, (2) such interpretation does not match with k (i.e., $interpretable_{a,i}(k')$), and (3) such 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 IV.3 (detected incorrect disambiguation)**.** One of our customers wanted to build a "Fitness Tamagochi", a game where an avatar grows depending on how much workout the user does. The customer told: *"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, where k' is both interpretable and acceptable, nocuous unconscious disambiguation phenomena [16] are likely to occur. Consider as example the case where 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 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 has been committed to a requirement document. We refer to these cases as *undetected incorrect disambiguation*. In our interviews we have been 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 Ex. III.4, where 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 IV.4 (undetected incorrect disambiguation)**.** In one of our interviews, the customer wanted to have an automated baby swinger. She told: *"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 behaviour of the system from the user interface point of view, the customer told: *"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 utterance of the customer showed an inconsistency with respect to the requirements understood by the analyst (i.e., $\neg acceptable^R_{a,i}(k)$).

V. Disclosing Tacit Knowledge

Ambiguities in requirements elicitation interviews are strictly interwoven with the concept of *tacit knowledge* [9], [32], [37]–[39]. In the reminder this section, we will analyse this relationship and suggest how to use ambiguity as a tool to disclose tacit knowledge.

A. Ambiguity and Tacit Knowledge

To better understand the connection between ambiguity and tacit knowledge, we rely again on the framework for tacit

knowledge illustrated in [32]. The framework considers four different classes of knowledge that play a role in requirements elicitation:

- *known known*, which identifies relevant information that is successfully passed from the customer to the analyst (i.e., it is known for both of them);
- *known unknown*, as relevant information that has not been 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 (is known to the customer), but does not pass to the analyst, and the analyst is not aware of the existence of such information (is unknown to the analyst);
- *unknown unknown*, as information that is relevant for the system to develop, but is unknown to both the customer and the analyst;

Some examples of these phenomena can be found in [9]. Here, we will contribute with examples taken from our direct experience.

Example V.1 (known unknown)**.** As an example of *known unknown*, consider the case of one of our interviews, where the customer, a domain expert in Public Administration, wishes to realise 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 to the current process adopted for monitoring such projects, and will ask questions concerning the process. Such knowledge is part of the known unknown dimension.

Example V.2 (unknown known)**.** As *unknown known* example, we can consider again the case of the History of Arts expert. The customer knew that pictures of paintings are available in specialised archives called photo libraries, but, in many of the cases considered interesting for the customer, are in paper format and *not* digitalised. 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 if the information was important – responded: *"Well, a digital archive does not exist!"*.

Example V.3 (unknown unknown)**.** *Unknown unknown* phenomena might occur, for example, in case of 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* (i.e., a relevant information that can potentially pass from the customer to the analyst) into an *known unknown* in practice, i.e., an information that the customer holds, but it is not successfully accessed by the analyst, although 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 occur, will not be articulated again by the customer.

Ambiguities are also a *resource*, since an ambiguity might appear in situations when the customer cannot articulate an idea properly. In some cases, this might happen because some domain knowledge could be hidden in his mind in the form of procedural knowledge [40], 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 further clarifications and possibly opening new directions for scoping the problem.

B. Disclosing Tacit Knowledge through Ambiguity

We have previously introduced the two predicates $articulated_{c,i}(k)$, and $accessible_{a,i}(k)$ (see Sect. III). The framework in [32] 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, s/he articulates a unit of information k that has some link in his 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 Ex. III.1, where 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 to 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 (Ex. III.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 Ex. IV.3, where a *detected incorrect disambiguation* case occurred, and in Ex. IV.1, where a *multiple understanding* situation is shown.

Another representative case is Ex. III.3, where 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, Ex. III.5 shows that the requirement of the customer was associated to 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 clarifying 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, either related to requirements (Ex. III.2, IV.3, IV.1), to goals (Ex. III.3) or to the domain (Ex. III.5, III.1). According to such rationale, we have inspected the fragments that caused ambiguity in our interviews, and, through joint discussions among the authors, we have considered whether such cases helped in disclosing tacit knowledge. The result of this analysis is shown in Fig. 2. We see that only in 8% of the cases the ambiguity did not lead to disclosing some type of knowledge. These were the cases where the ambiguity was either not resolved, or caused by babbling [32], or inapplicable technological expectations of the customer. Moreover, we see that also a certain percentage of unknown unknown cases (domain unk unk 1%, and requirements unk unk 12%) could be accessed. An example in this group concerns the swim keeper in Ex. III.6, where a previously unknown unknown requirement (i.e., the need to detect when the swimmer changes direction) emerged thanks to an *acceptance unclarity*. In it worth highlighting that, since the evaluation has been performed by the authors, these results have to be considered as part of the vision presented, and not as an empirical confirmation of such vision.

The only cases where tacit knowledge cannot be accessed, are those of *undetected incorrect disambiguation*. Indeed, except for those situations where another ambiguity helps in discovering the problem (see Ex. III.4 and Ex. IV.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:

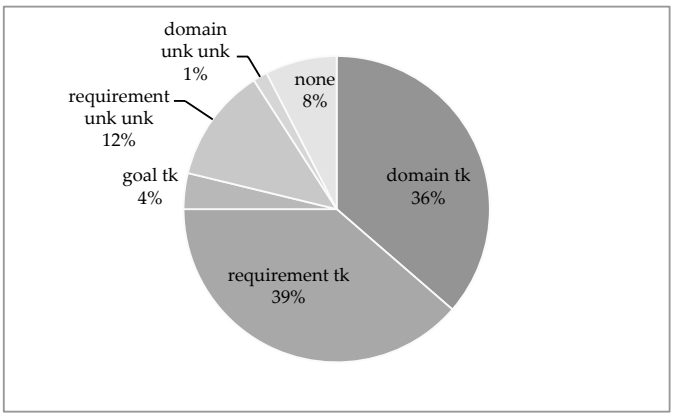

Fig. 2: Types of knowledge discovered during the interviews (tk = tacit knowledge, unk unk = unknown unknown).

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

Of course, this finding relies on the need to have $articulated_{c,i}(k)$, with k associated to 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 [9].

VI. 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 analyse four main challenges and some lesson learnt about them during our on-the-field experience.

Challenge 1: Identification of Ambiguity Cues

In our interviews, 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., [15], [16], [18], [19]) can provide a first help, as shown in Ex. III.2, where 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 Ex. III.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., [41], [42]). At the conversation level, analogous goals should be achieved also for oral interviews.

Term-based cues can also be associated to a peculiar phenomenon of requirements interviews that we have observed,

and that we call *meaning migration*. When a novel system has to be developed, novel terminology is needed to describe its structure and behaviour. Therefore, the customers tend to use common terms to denote system-specific entities or actions. The term "tap" in Ex. III.4 is an example of this phenomenon, where a term *migrates* from the vocabulary of the customer to the terminology of the system. Identifying such type of cues is not easy, and requires further research. From our experience, we conjecture 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 such term in conjunction with the other terms used) can quantify the potential novelty – and, hence, ambiguity – of such 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 to the usage of specific terms (see, e.g., Ex. III.5 and Ex. III.6), and research has to go beyond term-based cues. In such situations, pragmatic cues at the level of discourse have to be identified, which take into account the pragmatic facet described in Sect. II-C. In NL requirements, we have studied methods to detect pragmatic ambiguities [29], [30] 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 dialogues [34].

Challenge 2: Ambiguity-based Elicitation Methodologies

To profitably employ ambiguity in interviews, elicitation methodologies that leverage ambiguities have to defined and integrated with existing practices [9]. Such methodologies should take into account the fact that the analyst has to be trained in ambiguity detection, keep their 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 have experienced 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 he/she terminated his/her discourse. Another good practice was performing intermediate recapitulation of the problem to the customer. Rephrasing is indeed recognised as a powerful strategy to achieve a shared understanding [43], [44]. In our experiments, we have been able to detect cases of incorrect disambiguation specifically thanks to rephrasing. Moreover, we have seen that performing intermediate recapitulations helps establishing a shared terminology, and gently drive the customer towards *algorithmic mindset* and lexicon, where terms have a precise meaning, and system goals and behaviours are sharply defined. In particular, during the interviews with the History of Arts and Public Administration expert, the recapitulations let gradually emerge a sort of "pidgin" (i.e., a common language) [45] where the terms used came from the domain of the customer, but the discourse structure was more computer-science oriented.

Challenge 3: Ambiguity in the Process

Requirements collected during elicitation have an evolution along the whole software life-cycle. After eliciting the requirements, these are normally committed to a document, most of the time in NL [46]. In textual NL requirements, 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 emerged in elicitation cannot be properly expressed in written language. On the other hand, the precision could also lead to over-confidence of the requirements analyst on the absence of ambiguity, and its presence could be covered by the more formal surface of 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 the different stages of development. To this end, a phenomenology of ambiguity, similar to the one presented in this paper, should be defined at the level of editing, negotiating, refining and testing requirements. Such 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, which is more concerned with the input/output relation rather than with the 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.

Challenge 4: Ambiguity on the Customer 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 modelled. In our experience, as for the analyst many ambiguities are triggered by domain-specific terms, for the customer ambiguity is often raised 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 [33] is a primary goal in interviews. Like analyst-perceived ambiguity can disclose tacit knowledge of the customer, we conjecture 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 [47] 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.

VII. Conclusion

In this paper we have presented a phenomenology of ambiguity in requirements elicitation interviews, and we have stressed the primary role of ambiguity in disclosing tacit knowledge. Our vision is based on the convinction, in line with the arguments in [25] and [31], that ambiguity is a subjective phenomenon, which derives from the relation between

the signifier (i.e., speech fragment, word or sentence) and the reader/listener, who assigns a meaning to the signifier. Moreover, our view shows that ambiguity is also a contextual and situational phenomenon, where the pragmatic facet of the recipient of the signifier plays a primary role. 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 give a meaning to what they perceive [5]. Our near future commitment is to provide conceptual models to deeply understand the phenomenon. Such models will enable analysts to achieve a conscious skeptical attitude [43], [44], to go beyond the surface meaning of what they hear, and ultimately access the unknown.

ACKNOWLEDGMENT

The authors would like to thank Mattia Fazzini, for his support in coordinating the interviews, and all the people playing the role of customers. This work was partially supported by the LearnPAd FP7-ICT-2013.8.2 European Project.

REFERENCES

[1] I. Sommerville and P. Sawyer, "Viewpoints: principles, problems and a practical approach to requirements engineering," *Annals of Software Engineering*, vol. 3, no. 1, pp. 101–130, 1997.
[2] A. Distanont, H. Haapasalo, M. Vaananen, and J. Lehto, "The engagement between knowledge transfer and requirements engineering," *IJKL*, vol. 1, no. 2, pp. 131–156, 2012.
[3] S. Robertson and J. Robertson, *Mastering the requirements process: Getting requirements right*. Addison-wesley, 2012.
[4] D. Zowghi and C. Coulin, "Requirements elicitation: A survey of techniques, approaches, and tools," in *Engineering and managing software requirements*. Springer, 2005, pp. 19–46.
[5] I. Hadar, P. Soffer, and K. Kenzi, "The role of domain knowledge in requirements elicitation via interviews: an exploratory study," *Requir. Eng.*, vol. 19, no. 2, pp. 143–159, 2014.
[6] R. Agarwal and M. R. Tanniru, "Knowledge acquisition using structured interviewing: an empirical investigation," *JMIS*, vol. 7, no. 1, pp. 123–140, 1990.
[7] G. J. Browne and M. B. Rogich, "An empirical investigation of user requirements elicitation: Comparing the effectiveness of prompting techniques," *JMIS*, vol. 17, no. 4, pp. 223–249, 2001.
[8] W. R. Friedrich and J. A. Van Der Poll, "Towards a methodology to elicit tacit domain knowledge from users," *IJIKM*, vol. 2, no. 1, pp. 179–193, 2007.
[9] A. Sutcliffe and P. Sawyer, "Requirements elicitation: towards the unknown unknowns," in *RE'13*. IEEE, 2013, pp. 92–104.
[10] A. Davis, O. Dieste, A. Hickey, N. Juristo, and A. M. Moreno, "Effectiveness of requirements elicitation techniques: Empirical results derived from a systematic review," in *RE'06*. IEEE, 2006, pp. 179–188.
[11] J. Coughlan and R. D. Macredie, "Effective communication in requirements elicitation: a comparison of methodologies," *Requir. Eng.*, vol. 7, no. 2, pp. 47–60, 2002.
[12] A. M. Hickey and A. M. Davis, "A unified model of requirements elicitation," *J. Manage. Inf. Syst.*, vol. 20, no. 4, pp. 65–84, Mar. 2004.
[13] C. F. Alves, S. Pereira, G. Valença, J. Pimentel, and R. V. de Andrade, "Preliminary results from an empirical study in market-driven software companies." in *WER'07*, 2007, pp. 127–134.
[14] P. G. Neumann, "Only his only grammarian can only say only what only he only means," *ACM SIGSOFT SE Notes*, vol. 9, no. 1, p. 6, 1986.
[15] D. M. Berry and E. Kamsties, "The syntactically dangerous all and plural in specifications," *IEEE Software*, vol. 22, no. 1, pp. 55–57, 2005.
[16] D. M. Berry, E. Kamsties, and M. M. Krieger, "From contract drafting to software specification: Linguistic sources of ambiguity," 2003.
[17] L. Mich and R. Garigliano, "Ambiguity measures in requirements engineering," in *ICS'00, 16th IFIP WCC*, 2000, pp. 39–48.
[18] N. Kiyavitskaya, N. Zeni, L. Mich, and D. M. Berry, "Requirements for tools for ambiguity identification and measurement in natural language requirements specifications," in *WER'07*, 2007, pp. 197–206.
[19] S. Gnesi, G. Lami, and G. Trentanni, "An automatic tool for the analysis of natural language requirements," *IJCSSE*, vol. 20, no. 1, 2005.
[20] W. M. Wilson, L. H. Rosenberg, and L. E. Hyatt, "Automated analysis of requirement specifications," in *ICSE'97*, 1997, pp. 161–171.
[21] L. Mich, "NL-OOPS: from natural language to object oriented requirements using the natural language processing system LOLITA," *NLE*, vol. 2, no. 2, pp. 161–187, 1996.
[22] V. Ambriola and V. Gervasi, "On the systematic analysis of natural language requirements with Circe," *ASE*, vol. 13, 2006.
[23] L. Kof, "From requirements documents to system models: A tool for interactive semi-automatic translation," in *RE'10*, 2010.
[24] B. Gleich, O. Creighton, and L. Kof, "Ambiguity detection: Towards a tool explaining ambiguity sources," in *REFSQ'10*, ser. LNCS, vol. 6182. Springer, 2010, pp. 218–232.
[25] F. Chantree, B. Nuseibeh, A. N. D. Roeck, and A. Willis, "Identifying nocuous ambiguities in natural language requirements," in *RE'06*, 2006, pp. 56–65.
[26] A. Cimatti, M. Roveri, A. Susi, and S. Tonetta, "Formalizing requirements with object models and temporal constraints," *SoSyM*, vol. 10, no. 2, 2011.
[27] H. Yang, A. De Roeck, V. Gervasi, A. Willis, and B. Nuseibeh, "Extending nocuous ambiguity analysis for anaphora in natural language requirements," in *RE'10*. IEEE, 2010, pp. 25–34.
[28] H. Yang, A. N. D. Roeck, V. Gervasi, A. Willis, and B. Nuseibeh, "Analysing anaphoric ambiguity in natural language requirements," *Requir. Eng.*, vol. 16, no. 3, pp. 163–189, 2011.
[29] A. Ferrari, G. Lipari, S. Gnesi, and G. O. Spagnolo, "Pragmatic ambiguity detection in natural language requirements," in *AIRE'14*. IEEE, 2014, pp. 1–8.
[30] A. Ferrari and S. Gnesi, "Using collective intelligence to detect pragmatic ambiguities," in *RE'12*. IEEE, 2012, pp. 191–200.
[31] A. K. Massey, R. L. Rutledge, A. I. Anton, and P. P. Swire, "Identifying and classifying ambiguity for regulatory requirements," in *RE'14*. IEEE, 2014, pp. 83–92.
[32] V. Gervasi, R. Gacitua, M. Rouncefield, P. Sawyer, L. Kof, L. Ma, P. Piwek, A. De Roeck, A. Willis, H. Yang *et al.*, "Unpacking tacit knowledge for requirements engineering," in *Managing requirements knowledge*. Springer, 2013, pp. 23–47.
[33] M. Glinz and S. A. Fricker, "On shared understanding in software engineering: an essay," *CSRD*, pp. 1–14, 2014.
[34] M. Corvera Charaf, C. Rosenkranz, and R. Holten, "The emergence of shared understanding: applying functional pragmatics to study the requirements development process," *ISJ*, vol. 23, no. 2, pp. 115–135, 2013.
[35] M. G. Pitts and G. J. Browne, "Stopping behavior of systems analysts during information requirements elicitation," *J. Manage. Inf. Syst.*, vol. 21, no. 1, pp. 203–226, Jan. 2004.
[36] A. van Lamsweerde, *Requirements Engineering - From System Goals to UML Models to Software Specifications*. Wiley, 2009.
[37] M. Polanyi, *The Tacit Dimension*. Garden City, NY: Doubleday, 1966.
[38] N. Maiden and G. Rugg, "ACRE: selecting methods for requirements acquisition," *Soft. Eng. Journal*, vol. 11, no. 3, pp. 183–192, 1996.
[39] R. Gacitua, L. Ma, B. Nuseibeh, P. Piwek, A. De Roeck, M. Rouncefield, P. Sawyer, A. Willis, and H. Yang, "Making tacit requirements explicit," in *MARK'09*. IEEE, 2009, pp. 40–44.
[40] B. Kogut and U. Zander, "Knowledge of the firm, combinative capabilities, and the replication of technology," *Organization science*, vol. 3, no. 3, pp. 383–397, 1992.
[41] A. Ferrari, F. dell'Orletta, G. O. Spagnolo, and S. Gnesi, "Measuring and improving the completeness of natural language requirements," in *REFSQ'14*, ser. LNCS, vol. 8396. Springer, 2014, pp. 23–38.
[42] R. Gacitua, P. Sawyer, and V. Gervasi, "Relevance-based abstraction identification: technique and evaluation," *Requir. Eng.*, vol. 16, no. 3, pp. 251–265, 2011.
[43] N. Karten, *Managing Expectations: Working with People Who Want More, Better, Faster, Sooner, NOW!* Addison-Wesley, 2013.
[44] H. Saiedian and R. Dale, "Requirements engineering: making the connection between the software developer and customer," *Information and Software Technology*, vol. 42, no. 6, pp. 419–428, 2000.
[45] P. Mühlhäusler, *Pidgin and creole linguistics*. Blackwell Oxford, 1986.
[46] L. Mich, M. Franch, and P. N. Inverardi, "Market research for requirements analysis using linguistic tools," *Requir. Eng.*, vol. 9, no. 1, pp. 40–56, 2004.
[47] H. H. Clark, *Using language*. Cambridge University Press, 1996.