Ensuring Action: Identifying Unclear Actor Specifications in Textual Business Process Descriptions

Ulf Sanne[1], Hans Friedrich Witschel[1], Alessio Ferrari[2] and Stefania Gnesi[2]
[1]*Fachhochschule Nordwestschweiz, Riggenbachstr. 16, 4600 Olten, Switzerland*
[2]*ISTI, CNR, Via G. Moruzzi 1, 56124 Pisa, Italy*

Keywords: Business Process Management, Quality Assessment, Natural Language Processing.

Abstract: In many organisations, business process (BP) descriptions are available in the form of written procedures, or operational manuals. These documents are expressed in informal natural language, which is inherently open to different interpretations. Hence, the content of these documents might be incorrectly interpreted by those who have to put the process into practice. It is therefore important to identify language defects in written BP descriptions, to ensure that BPs are properly carried out. Among the potential defects, one of the most relevant for BPs is the absence of clear actors in action-related sentences. Indeed, an unclear actor might lead to a missing responsibility, and, in turn, to activities that are never performed. This paper aims at identifying unclear actors in BP descriptions expressed in natural language. To this end, we define an algorithm named ABIDE, which leverages rule-based natural language processing (NLP) techniques. We evaluate the algorithm on a manually annotated data-set of 20 real-world BP descriptions (1,029 sentences). ABIDE achieves a recall of 87%, and a precision of 56%. We consider these results promising. Improvements of the algorithm are also discussed in the paper.

1 INTRODUCTION

In several contexts, which range from private companies to public administrations, business process (BP) descriptions are available in natural language. Indeed, although more formal graphical notations have emerged to model BPs, such as BPMN (Business Process Modelling and Notation) or YAWL (Yet Another Workflow Language), most of the legacy process knowledge – when not *tacit* – is still conveyed in paper-like documents, which have the form of procedures or operational manuals. In addition, even when graphical models are available, these are often complemented by textual descriptions (Schumann et al., 2014). Indeed, as noted by Ottensooser et al. (Ottensooser et al., 2012) and by Navrocki et al. (Nawrocki et al., 2006), the understandability of a BP model is higher when complemented with text. On the other hand, given the informal nature of natural language, textual descriptions might be *unclear*. In particular, Sommerville highlights that, if a process description does not assign a clear responsibility for tasks that are part of the process – i.e., if the *actor* is unclear – this might result in several organizational vulnerabilities (Sommerville, 2007), namely: (a) unassigned responsibility – i.e., the task is not performed, since nobody is in charge; (b) duplicated responsibility – i.e., the task is performed by more than one actor, with duplicated effort; (c) uncommunicated responsibility – i.e., the task remains undone, since the actor in charge of the task is not aware of his/her responsibility. Although this problem might be addressed with the introduction of, e.g., BPMN models, in which activity icons and swim lanes support the specification of actors, the problem might remain in the text that complements such models.

Several studies were performed in the literature to improve the quality of BP descriptions expressed as models, to ensure their correctness (Morimoto, 2008), and to improve their understandability (Reijers and Mendling, 2011). Furthermore, studies were also performed to identify defects in the textual labels of BP models (Leopold et al., 2013), and to generate textual descriptions from BP models (Leopold et al., 2014). However, none of the studies addresses the problem of the quality of the description of BP written by human editors.

This paper aims at filling this research gap. In particular, we focus on the detection of *unclear actors* in BP specifications written in natural language. To

Sanne, U., Witschel, H., Ferrari, A. and Gnesi, S.
Ensuring Action: Identifying Unclear Actor Specifications in Textual Business Process Descriptions.
DOI: 10.5220/0006040301400147
In *Proceedings of the 8th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2016) - Volume 3: KMIS*, pages 140-147
ISBN: 978-989-758-203-5

this end, we designed and implemented an algorithm named ABIDE (unclear Actor detection in BusIness process DEscription). The algorithm leverages a set of heuristics, and makes use of rule-based natural language processing (NLP) techniques to identify statements with unclear actors, including cases of *missing* actor – i.e., when the actor is not specified and thus a case of unassigned or uncommunicated responsibility might result –, *meaningless* actor – i.e., the term that identifies the actor does not have sense in the context of the document (again, unassigned responsibility may be the result) –, and *ambiguous* actor – i.e., the term that identifies the actor can be interpreted in different ways, which might lead to duplicated responsibility. To evaluate ABIDE, we employed a set of 20 real-world BP descriptions (1,029 sentences), which were previously annotated for clarity defects by human operators. Then, we defined two classical baseline algorithms – never warn and warn randomly – against which we compared the performance of ABIDE in predicting the manual annotations. ABIDE outperforms the two baselines, and achieves a recall of 87% and a precision of 56% on the data. To our knowledge, this is the first work that addresses the problem of unclear actors in BP descriptions, and we consider these results a promising starting point.

2 RELATED WORK

Our research is related to research in the area of description of business processes – in particular regarding the assignment of roles and responsibilities – and to research on clarity of natural language descriptions in general.

2.1 Principles for Describing Responsibilities in Business Processes

Formal notations for the description of business processes include means to express responsibility for executing an action – for instance, BPMN (OMG, 2011) foresees *swimlanes* to represent roles or actors responsible for any activity. In military orders, the 5-W principle is used to describe an action (Lind and Lubera, 2009), including, besides the *what*, *where*, *when* and *why* also the *who*, i.e. who is responsible for an action.

Concerning the assignment of responsibilities, previous research has distinguished different types of responsibilities. The so-called RACI charts (Smith and Erwin, 2005) summarize the common ways how people can be involved in a particular activity. The abbreviation RACI stands for responsible, accountable, consulted and informed. In our work, we concentrate on *responsibility*. Indeed, from the business process perspective, consulted persons are not actors, but resources used in activities (Ciabuschi et al., 2012). Informed actors do not actively contribute, which turns them into stakeholders rather than actors (Voinov and Bousquet, 2010). Finally, accountability is rather a legal than a functional term.

Within RACI, the definition of "responsible" is as follows: "The actor technically responsible. It means that he or she is in charge of carrying out the activity under given circumstances and with given means and resources. Responsibility can be shared." In our research, we investigate to what extent responsibility for an activity – as given by this definition – is clearly defined by a textual BP description.

2.2 Text Clarity

There is extensive literature that promotes principles ensuring clarity, conciseness and the absence of technical jargon in written communication. Examples of such literature include *The Plain English Guide* (Cutts, 1996) or the reference book *Style: Toward Clarity and Grace* (Williams and Colomb, 1995) – containing practical examples and guidelines of how to write clearly.

In terms of clarity defects, **ambiguity** has been studied extensively in scientific work. Ambiguity of *terms* is an open problem in the computational linguistic community, and is traditionally associated to the so-called word-sense disambiguation (WSD) task (Navigli, 2009; Ide and Véronis, 1998). Several approaches exist that address this problem, which use unsupervised (Agirre and Edmonds, 2007; Véronis, 2004), supervised (Lee and Ng, 2002) and knowledge-based approaches (Banerjee and Pedersen, 2003; Navigli and Velardi, 2005).

Ambiguity as a quality defect has been largely studied in the field of requirements engineering. In software engineering, requirements need to be understood by different stakeholders and should be as little ambiguous as possible to avoid misunderstanding among the stakeholders. Therefore, several studies have been performed to categorise and detect ambiguities in NL requirements.

Part of the works are focused on the identification of typical ambiguous terms and constructions (Berry and Kamsties, 2005; Berry et al., 2003; Gnesi et al., 2005; Wilson et al., 1997; Gleich et al., 2010). One of the seminal works on ambiguity in requirements is the

one of Berry *et al.* (Berry et al., 2003), which has been implemented in tools such as QuARS (Gnesi et al., 2005) and ARM (Wilson et al., 1997) – which detect *lexical* ambiguities, based on vague, weak or subjective expressions (e.g., "as soon as possible", "reasonably"). Another research direction tries to translate text into some kind of formal representation in order to automatically detect problems with its interpretation (Ambriola and Gervasi, 2006; Kof, 2010). Work on syntactic ambiguity in requirements has focused on *anaphoric* (e.g., (Yang et al., 2011)) – i.e., associated to the interpretation of pronouns – and *coordination* ambiguities (e.g., (Chantree et al., 2006)) – i.e., associated with coordinating conjunctions.

To the best of our knowledge, no research has been conducted to address quality defects related to the (lack of) clarity of actor specifications in textual business process descriptions. For the reasons outlined above, we consider this a relevant gap that our research tries to close.

3 DATA-DRIVEN PROBLEM AWARENESS

In order to better understand "actor unclear" defects in textual business process descriptions, we first compiled a corpus consisting of documents that describe procedures in public administrations.

In order to select the documents, we first identified websites that include pointers to publicly accessible BP descriptions (e.g., the US Nuclear Commission Website [1], the UK Health and Safety Website [2], the US Court Website [3]) and then selected a set of 20 documents from those Websites, containing a total of 1,029 sentences. In doing so, we excluded documents requiring special expertise (e.g. that of a lawyer), as well as high-level regulations.

We then recruited 17 annotators and made sure that all documents from the data set were annotated by at least two annotators. We instructed the annotators by means of examples of defective sentences and told them to tag in a sensitive way, i.e. be rather strict in tagging sentences as defective. The tagging resulted in 126 sentences being annotated as having an *unclear actor*. This may only occur when the sentence describes a BP activity, i.e., something that would be translated into an activity shape in BPMN, such as a Task or a Process. Hence, our first objective was to identify which were these types of sentences, and in which way they differed from other types of sentences. By manually analysing the data-set, we identified 8 types of sentences in the textual BP descriptions. **Activity sentences** describe an instruction to be performed by some actor involved in the BP. Example: *The LEAP Academy employee will submit to the Commissioner of Education an enrollment report for the forthcoming year by June 1.* **Business rules** ban or enforce actions or results of actions. Example: *The EPBR application shall adhere to the template in annex A.* **Motivational statements** explain the purpose or goal of a process. Example: *The formal second opinion (FSO) procedure is an important part of ensuring that OSD maintains high standards of assessment decision-making.* **Introductory summaries** summarize briefly a set of activities and other process features in advance, without claiming to comprise all necessary information. Example: *After submission, each application will be subject to a two-step selection procedure.* **Repeating summaries** provide a brief repetition of what has been explained before, to reinforce the reader's memory or highlight important aspects. Example: *After completion of all steps described above, the project manager has now established a full list of relevant stakeholders end their concerns in the project.* **Activity meta-information** constitutes additional information about an activity, which does not belong to the activity description itself. Example: *The required realtime coordination of the distributed deployment team is possible, since the team is equipped with mobile communication devices.* **Background information** describes background or context. Example: *The City of Austin has established a major event initiative that is supported by the Police Department, EMS and Fire Department.* **Definitions** explain a term for later use. Example: *The Service Conference is a meeting in which the applicant and the other parties involved discuss about the application.*

For our task, we considered only the sentences of the first type, since they were those that could potentially include an unclear actor defect. Hence, we manually identified activity sentences in the data-set. The analysis resulted in 255 sentences, including the previously annotated unclear actor defects (126 sentences). From now on, this annotated set of sentences will be referred to as the *gold standard*.

We then analysed the defects that had been annotated in order to understand which categories of problems exist and what solutions might help to resolve them. The result of this analysis was the identification of three main problem classes, namely:

[1] http://www.nrc.gov/about-nrc/policy-making/internal.html

[2] http://www.hse.gov.uk/foi/internalops/

[3] http://www.uscourts.gov/rules-policies/current-rules-practice-procedure

- **Missing Actor:** the sentence does not include any explicit actor;
- **Meaningless Actor:** the sentence includes an explicit actor, but a human reader may not understand what is meant by the term used to identify the actor;
- **Ambiguous Actor:** the sentence includes an explicit actor, but there is more then one way to understand the meaning of the term used to describe the actor.

Examples for each class are reported in Table 1. The different heuristics that compose ABIDE are designed to address these classes of defects.

4 A RULE-BASED ALGORITHM FOR DEFECT DETECTION

4.1 NLP Technologies Adopted

Before describing the heuristics that we defined to identify the defects, it is useful to discuss shortly the natural language processing (NLP) technologies that we adopted to extract information from the documents, and that will be referred in the following sections: The preprocessing starts with a **sentence segmentation** of the text, followed by **tokenization**, i.e. partitioning of the text into separate tokens, such as words, numbers and punctuation. Next, **Part-of-Speech (POS) Tagging** is performed which associates to each token a Part-of-Speech, e.g., noun (NN), verb (VB), adjective (JJ), *etc.* The POS tagging forms the basis for a shallow parsing that identifies noun phrases (NP, "noun chunking") and verb phrases (VP, "verb chunking") in sentences. This will allow later to identify e.g. chunks that refer to actors. Finally, we apply a **Gazetteer** which searches for occurrences of terms defined in a list of terms. It can be used to check for e.g. the presence of vague terms in the documents.

Based on these preprocessing steps, out heuristics were implemented within the tool GATE (General Architecture for Text Engineering (Cunningham, 2002)) in the form of so-called **JAPE Rules**. Such rules allow defining high-level regular expressions over tokens and other elements in a text. They identify *patterns* of elements that match the rule. Since JAPE rules can be rather long to report, we will use a more concise and intuitive pseudo-code to present the heuristics which is inspired by the JAPE grammar.

In JAPE, and in our rules, we use the usual symbols from the syntax of regular expressions to express e.g. logical conjunction or disjunction.

4.2 Heuristics for Missing Actor

The first heuristics that we describe allows to identify sentences in which the actor is missing. For sentences in *active* form, an actor is missing only when a verb in imperative form is used, e.g., in *Delete the application if the two-months period has expired.* However, in these cases, the actor is expected to be the reader of the sentence. Hence, we do not consider these situations as cases in which the actor is missing. For sentences in *passive* form, the actor is missing when the sentence does not include a "by" clause to express a subject, e.g., *The procedure shall be carried out before the end of March 2015.*

The former type of sentences are all those sentences that include the following pattern:

$$P_{MIS} = (Token \in Aux)\ (Token.POS == VBN|VBD)(Token)* \ (\neg \text{“by”}) \quad (1)$$

The pattern matches any case in which we have a term that indicates the presence of at least an auxiliary verb ($Token \in Aux$, i.e.,"am", "are", "were", "being", "is", "been", "was", "be") followed by a past participle (VBN) or past tense (VBD). Moreover, the rule checks the absence of the Token "by" in the same sentence which is an indicator of the potential specification of an actor. The notation $(Token)*$ indicates that the verb might be followed by zero or more Tokens, before the Token "by" is found. All the sentences including the previous pattern are marked as *Defective* by ABIDE.

4.3 Heuristics for Meaningless Actor

Even when a sentence includes an actor, the term used to name the actor might not be understandable by the reader, i.e., the actor is *meaningless*. To identify sentences with meaningless actors, ABIDE uses the following heuristics. First ABIDE searches for potential actors in the sentence. To this end, the algorithm extracts subject-verb-object (SVO) triples from the sentence, and names as potential actors all the nouns playing the role of *subject* in the sentence. Then, ABIDE checks whether all the potential actors expressed in the sentence can be understood by the reader. This is done by examining whether each of the actors candidates in the sentence belongs to a dictionary of terms and whether it is not an acronym. In particular, in our implementation, the algorithm checks a) whether the term can be found in Wikipedia and b) whether it is an acronym, i.e. consists of all upper case characters. The rationale of this approach is the following: in the case of a), although the reader

Table 1: Sub-classes of the actor unclear problem.

Problem class	Description	Example(s)
Actor missing	An activity is described without referring to an actor	The request for purchase form will be forwarded to Purchasing (passive)
Actor meaningless	The term referring to an actor cannot be interpreted by the expected target audience	The *DCM* and *SCC* shall supply copies of relevant information [...]
Actor ambiguous	The way an actor is referred to can be interpreted in more than one way	Finally, they must be dated, and signed by *the relevant person within the institution* [...]

does not necessarily know the meaning of the term expressing the actor, he/she can access Wikipedia and associate a meaning to the actor. For b), we assume that readers may not be familiar with acronyms if they are not previously introduced in the text (or even then might quickly forget their meaning) – our algorithm does not check previous introduction of the full form of acronyms, an extension that may be added as future work.

To extract SVO triples, the algorithm leverages shallow parsing, and checks each sentence for the following pattern:

$$P_{SVO} = (NP)(VP)(NP) \tag{2}$$

The pattern matches any triple in which we have a noun chunk followed by a verb chunk and by a noun chunk. The first noun chunk is expected to include the subject of the sentence. However, it might be composed of more than one Token, as e.g., *The principal HDEC*. For all the nouns in the first NP in P_{SVO} – referred as *Subject* in the following – the algorithm checks whether the noun can be found in Wikipedia. To this end, the following pattern is applied:

$$\begin{aligned} P_{UNK} = (&Token.POS =\sim NN*, \\ &Token \in Subject, \\ &Token \notin Wikipedia | Acronym(Token)) \end{aligned} \tag{3}$$

The pattern matches any Token representing a noun (i.e., all Tokens which have a POS starting with NN[4]), which is included in *Subject*, and that either does not belong to the Wikipedia dictionary or is recognised as an acronym – where $Acronym(\cdot)$ is a predicate that is true if a token consists only of upper case characters (possibly separated by periods). All the sentences including the previous pattern are marked as *Defective* by ABIDE.

4.4 Heuristics for Ambiguous Actor

If the actor has a meaning that can be found in Wikipedia, this does not imply that the actor is not ambiguous. Hence, we define three additional heuristics to check for ambiguous actors. Three main cases of ambiguous actor are identified by ABIDE:

1. **Ambiguous Noun:** an actor might be ambiguous if the term that identifies the actor can have different meanings in different linguistic contexts. For example, the term *Assessor* can be the assistant to a judge or magistrate, in a legal context, and or an an expert who calculates the value of property, in the real-estate appraisal domain.
2. **Ambiguous Pronoun:** an actor might be ambiguous if a pronoun – e.g., he, it, him, her – is used to refer to more than one noun, as in the sentence: *The delegate assesses the presence of the candidate, and he provides his signature.* Here, the pronoun *he* can be referred to the *delegate* or to the *candidate*. These phenomena are normally called anaphoric ambiguities (Yang et al., 2011).
3. **Vague Modifier:** the name of an actor might be associated with a vague modifier as in *The relevant authority*, or *The proper office*.

To detect the cases described above, ABIDE leverages the *Subject* element extracted from the P_{SVO} pattern described in Sect. 4.3. In particular, to detect cases of ambiguous nouns (case 1), it looks up the *Subject* element in a list of ambiguous terms. For our experiments, we have compiled a list of such terms by analysing the previously annotated gold standard (see Section 3). We identified sentences describing an activity that had been tagged as defects and checked whether they contained unclear terms in their subjects. We included such terms in the list if we reckoned that they might occur in BP description across several domains. The current list is as follows: *AmbiguousTermList* = {“person”, “responsible”, “office”, “staff”, “employee”, “company”, “unit”, “those”, “all”, “somebody”, “team”}. Of course such a list – being derived from a fairly small corpus – cannot be claimed to be comprehensive. We believe that it can be extended and tuned for a domain when our approach is used in practice. Another way to extend it automatically would be to

[4]The notation $=\sim$ matches regular expressions

use bootstrapping approaches when descriptions are quality-checked and hence manually annnotated in a real-life setting. The following pattern summarises the approach to detect ambiguous nouns:

$$P_{AMB_N} = (Token.POS =\sim NN*, \quad Token \in Subject, \quad Token \in AmbiguousTermList) \tag{4}$$

To check for ambiguous pronouns (case 2), ABIDE simply checks whether the *Subject* element includes a pronoun. This might lead to false positive cases. However, more complex machine learning methods are required to handle anaphoric ambiguities (Yang et al., 2011), which can however leave some ambiguity undiscovered. To detect ambiguous pronouns, the following pattern is applied:

$$P_{AMB_P} = ((Token.POS == PP| \quad Token.POS =\sim PR*), \quad Token \in Subject) \tag{5}$$

The pattern matches any Token representing a personal pronoun (*PP*), or other types of pronouns (*PR*∗), which is included in *Subject*.

Finally to check for vague modifiers (case 3), ABIDE checks whether the *Subject* element includes one of the terms included in a list of vague modifiers. In our implementation, we used the list adopted by QuARS (Gnesi et al., 2005), which includes 446 vague terms. We refer this set with the name *Vague*. We first implemented the following pattern:

$$P_{AMB_V1} = (Token \in Vague, Token \in Subject) \tag{6}$$

After some first experiments with our data set, we realised that the relevance of the cases of vague modifiers was rather high, and that the low accuracy of the shallow parsers adopted was preventing ABIDE from finding cases of ambiguous subjects. We therefore relaxed our rule by using the following pattern:

$$P_{AMB_V2} = (Token \in Vague) \tag{7}$$

In the following, we only used the pattern P_{AMB_V2}. ABIDE marks as *Defective* any sentence that matches one of the following patterns: P_{AMB_N}, P_{AMB_P}, P_{AMB_V2}.

5 EVALUATION

To evaluate the ABIDE algorithm, we used the *gold standard* data set introduced in Section 3. It consists of 255 sentences that describe an activity within a business process. Of these, 126 were manually tagged as defective. We first describe the evaluation measures and baselines, then report results and finally analyse potential improvements.

5.1 Baselines and Evaluation Measures

Our goal was to run ABIDE on the gold standard and compare its annotations of defective sentences to the manual ones, using measures such as precision, recall and F-measure. In order to be able to judge the quality of these results, we compared them to two baseline taggers: **Baseline NONE** is a simple tagger that predicts no defect for every sentence. It corresponds to having no quality control for BP descriptions (a common approach in many organisations). **Baseline RANDOM** annotates a given sentence as defective with a given a priori probability p. In our case, since roughly 50% of all gold standard sentences are defective according to human judgement, we used $p = .5$. We also considered a "Baseline ALL", which would mark all sentences as defects – however, we conjecture that applying that baseline in practice will not lead to a perfect recall (i.e., to all defects being spotted) since the human who will do the quality assurance and who then has to look at every sentence will surely overlook defects. Since this effect is hard to quantify, we have not used such baseline in the experiment.

Standard measures such as precision and recall give equal weight to both types of mistakes that ABIDE can make – i.e., false positives and false negatives. In practice, the impact of a false positive can be substantially different from the impact of a false negative.

One way to take this difference into account is to use a version of the F-measure that places greater emphasis on, e.g., recall. Another option is a cost-based evaluation where one estimates the negative impact (cost) caused by each ABIDE decision. Figure 1 depicts the costs that we estimate to arise for our scenario.

		Actual Class	
		Defect	No Defect
Predicted Class	Defect	C	C
	No Defect	NC	0

Figure 1: A cost matrix for prediction of defects.

Whenever ABIDE predicts a defect, a warning is raised and a responsible person has to inspect the corresponding sentence. We assume that this causes an average loss of time (i.e., cost) of C minutes – the responsible has to re-consider the formulation of the sentence and sometimes possibly to clarify the situation.

When ABIDE fails to identify a defective sentence

Table 3: Categories of false positive passive sentences.

Error pattern	Example	Frequency
Passive in a part of the sentence which does not describe the activity	The FSO OM shall use the information to consider *if the acceptance criteria are met*.	18
X ensures that Y is done	The Senior Administrator [...] ensures that all required registration documentation is filed [...].	8
X is required/recommended/requested to do Y	Suppliers are expected to note interest in the contract and request the ITT documents.	4

(i.e. when a false negative occurs), the sentence remains in the final process description. We optimistically assume that process executors who read the sentence will not work on false assumptions, but will always spot the unclarity and attempt to clarify. Such clarification – that may involve speaking to colleagues or consulting other sources – causes a loss of time (cost) that we estimate to be at least as high as the above-mentioned cost C for handling raised warnings. Process descriptions usually have to be read – at least once – by all persons that are regularly involved in process execution. If the number of such persons is N, then, based on the above arguments, the cost of a false negative is at least NC. In our evaluation runs, we used $C = 1$ and $N = 5$ or $N = 10$, i.e. we assumed a situation with 5, respectively 10 process executors working based on a common process description.

5.2 Results

Table 2 shows the results of ABIDE and the two baselines in terms of precision, recall, F-measure and cost.

Table 2: ABIDE results.

Measure	ABIDE	NONE	RANDOM
Precision	0.56	1	0.5
Recall	0.87	0	0.5
F_1	0.69	0	0.5
Cost ($N = 5$)	285	630	448
Cost ($N = 10$)	355	1260	768

Although ABIDE is far from perfect in terms of precision, it clearly outperforms the baselines in terms of both F-measure and cost, indicating that a quality control based on ABIDE can help to save cost.

5.3 Qualitative Analysis

In order to derive potential future improvements of ABIDE's precision, we made an analysis of false positives, attempting to categorise them in terms of linguistic patterns. It turned out that interesting "false positive patterns" can be observed primarily in the area of passive sentences (see Section 4.2).

Table 3 shows the three categories of false positive passive sentences that we identified in the data, each with its frequency in our gold standard and an example.

In the first category, one often finds sentences where the activity to be performed is described in active voice, but – in the same sentence – e.g. a condition of that action is described in passive voice (as in the example given above in Table 3).

Since these error classes account for roughly one third of all false positives in our experiment, it might be worth extending ABIDE's rules, e.g. by checking whether passive is really used in the activity-related part of a sentence (category 1, first line in Table 3) or whether one of the patterns of category 2 or 3 (last two lines in Table 3) is present in a sentence.

6 CONCLUSIONS

In this paper, we have shown that it is feasible to build an algorithm that will support the quality control for business process descriptions in terms of how clearly they refer to responsible actors. We have implemented the algorithm ABIDE – based on some rather simple rule-based heuristics – that will detect sentences describing an activity, but with unclear or no reference to a responsible actor. We have shown that this algorithm outperforms the baselines in terms of its precision and recall and that it will help to save cost when applied in quality control.

In future, as indicated in Section 5.3, there are still several extensions and fine-tunings of ABIDE that we plan to address, in particular in the area of passive sentences, in which false positives may be avoided by considering some linguistic patterns that our qualitative analysis has revealed. We further plan to evaluate ABIDE in a real-life setting by discussing its results with persons responsible for writing and checking business process descriptions.

ACKNOWLEDGEMENT

This work is supported by the European Union FP7 ICT objective, through the Learn PAd Project with Contract No. 619583.

REFERENCES

Agirre, E. and Edmonds, P. G. (2007). *Word sense disambiguation: Algorithms and applications*, volume 33. Springer Science & Business Media.

Ambriola, V. and Gervasi, V. (2006). On the systematic analysis of natural language requirements with Circe. *ASE*, 13.

Banerjee, S. and Pedersen, T. (2003). Extended gloss overlaps as a measure of semantic relatedness. In *IJCAI*, volume 3, pages 805–810.

Berry, D. M. and Kamsties, E. (2005). The syntactically dangerous all and plural in specifications. *IEEE Software*, 22(1):55–57.

Berry, D. M., Kamsties, E., and Krieger, M. M. (2003). From contract drafting to software specification: Linguistic sources of ambiguity.

Chantree, F., Nuseibeh, B., Roeck, A. N. D., and Willis, A. (2006). Identifying nocuous ambiguities in natural language requirements. In *Proc. of RE'06*, pages 56–65.

Ciabuschi, F., Perna, A., and Snehota, I. (2012). Assembling resources when forming a new business. *Journal of Business Research*, 65(2):220–229.

Cunningham, H. (2002). GATE, a general architecture for text engineering. *Computers and the Humanities*, 36(2):223–254.

Cutts, M. (1996). *The plain English guide*. Oxford University Press.

Gleich, B., Creighton, O., and Kof, L. (2010). Ambiguity detection: Towards a tool explaining ambiguity sources. In *Proc. of REFSQ'10*, volume 6182 of *LNCS*, pages 218–232. Springer.

Gnesi, S., Lami, G., and Trentanni, G. (2005). An automatic tool for the analysis of natural language requirements. *IJCSSE*, 20(1).

Ide, N. and Véronis, J. (1998). Introduction to the special issue on word sense disambiguation: the state of the art. *Computational linguistics*, 24(1):2–40.

Kof, L. (2010). From requirements documents to system models: A tool for interactive semi-automatic translation. In *Proc. of RE'10*.

Lee, Y. K. and Ng, H. T. (2002). An empirical evaluation of knowledge sources and learning algorithms for word sense disambiguation. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 41–48. Association for Computational Linguistics.

Leopold, H., Eid-Sabbagh, R.-H., Mendling, J., Azevedo, L. G., and Baião, F. A. (2013). Detection of naming convention violations in process models for different languages. *Decision Support Systems*, 56:310–325.

Leopold, H., Mendling, J., and Polyvyanyy, A. (2014). Supporting process model validation through natural language generation. *Software Engineering, IEEE Transactions on*, 40(8):818–840.

Lind, H. and Lubera, M. (2009). Battle Management Language - An Implementation for a Military Scenario Editor.

Morimoto, S. (2008). A survey of formal verification for business process modeling. In *Computational Science–ICCS 2008*, pages 514–522. Springer.

Navigli, R. (2009). Word sense disambiguation: A survey. *ACM Computing Surveys (CSUR)*, 41(2):10.

Navigli, R. and Velardi, P. (2005). Structural semantic interconnections: a knowledge-based approach to word sense disambiguation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(7):1075–1086.

Nawrocki, J. R., Nedza, T., Ochodek, M., and Olek, L. (2006). Describing business processes with use cases. In *9th International Conference on Business Information Systems, BIS*, pages 13–27.

OMG (2011). Business Process Model and Notation (BPMN V 2.0).

Ottensooser, A., Fekete, A., Reijers, H. A., Mendling, J., and Menictas, C. (2012). Making sense of business process descriptions: An experimental comparison of graphical and textual notations. *Journal of Systems and Software*, 85(3):596 – 606.

Reijers, H. A. and Mendling, J. (2011). A study into the factors that influence the understandability of business process models. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 41(3):449–462.

Schumann, R., Delafontaine, S., Taramarcaz, C., and Evéquoz, F. (2014). Effective Business process documentation in federal structures. In *44. Jahrestagung der Gesellschaft für Informatik*, pages 1043–1057.

Smith, B. and Erwin, J. (2005). Role & Responsibility Charting (RACI).

Sommerville, I. (2007). *Models for responsibility assignment*, pages 165 – 186. Springer.

Véronis, J. (2004). Hyperlex: lexical cartography for information retrieval. *Computer Speech & Language*, 18(3):223–252.

Voinov, A. and Bousquet, F. (2010). Modelling with stakeholders. *Environmental Modelling & Software*, 25(11):1268–1281.

Williams, J. and Colomb, G. (1995). *Style: Toward Clarity and Grace*. Chicago guides to writing, editing, and publishing. University of Chicago Press.

Wilson, W. M., Rosenberg, L. H., and Hyatt, L. E. (1997). Automated analysis of requirement specifications. In *Proc. of ICSE'97*, pages 161–171.

Yang, H., Roeck, A. N. D., Gervasi, V., Willis, A., and Nuseibeh, B. (2011). Analysing anaphoric ambiguity in natural language requirements. *Requir. Eng.*, 16(3):163–189.