Multi-word Entity Extraction and Rich Relationship Identification to Derive Conceptual Models from Natural Language Specifications

Giuliana Maltempo[1], Juliana Delle Ville[1], Santiago Cecconato[1], Federico Pellegrino[1], Damiano Distante[2], and Leandro Antonelli[1,3]

[1] LIFIA, Fac. de Informática, UNLP, La Plata, Bs. As., Argentina
[2] Universita degli Studi di Roma
[3] CAETI - Facultad de Tecnología Informática - Universidad Abierta Interamericana

{gmaltempo, jdelleville, sceconato, lanto}@lifia.info.unlp.edu.ar
pampapellegrino@gmail.com
damiano.distante@unitelmasapienza.it

Abstract. Requirements engineering is a critical phase in software development. Errors in requirements specifications may become costly problems later on; therefore, such errors should be found and corrected early in the engineering process. Describing requirements in natural language is propitious for both the domain experts and the software development team. However, natural language may give rise to diverse interpretations as a consequence of the different backgrounds of the two participants involved. It is therefore necessary to provide guidance on the specification of unambiguous requirements. In previous work, we have advanced the notion of kernel sentences as an appropriate structure for the specification of knowledge. We have also discussed conceptual models as a useful technique to summarize specifications so that all participants have a concise overview of the domain. To achieve consistent and coherent specifications, we presented a two-step method: first compliance with kernel format is checked, and then a conceptual model is derived to summarize the knowledge gathered. This paper extends the conceptual model previously derived from kernel sentences by identifying multi-word entities and establishing various new relationships among entities. This is intended to help achieve better quality specifications. We also describe a prototype that uses natural language processing and artificial intelligence tools to support the method. Finally, we present the results of a preliminary evaluation of our method, which show a promising applicability.

Keywords: Requirements Specification, Kernel Sentences, Conceptual Model, Natural Language.

1 Introduction

Requirements engineering is a critical phase in software development. Errors arising from faulty requirements specifications could be 200 times more expensive to fix in later stages; therefore, such errors should be found and corrected as early as possible in the engineering process [5].

Two different groups of people take part in the requirements phase: clients (or domain experts), and the software development team. Clients state their needs and provide knowledge to be included in the software application. The development team must understand said needs and knowledge in order to provide the clients with a satisfactory software application. Clients and developers belong to different worlds and express themselves through different specialised languages [31]. The former group uses the language of the relevant domain (e.g., agriculture, finances, etc.), whereas the development team uses the jargon of computer science. Using natural language artifacts that are intelligible to both parties is the most suitable course of action for bridging this communicative gap [23]. However, natural language specifications are not free of defect.

Natural language may give rise to diverse interpretations, in part as a consequence of the different backgrounds of the participants involved in the requirements process [28]. It is therefore necessary to provide guidance on the specification of unambiguous requirements. Ambiguity is the possibility of attributing diverse interpretations to one expression. The source of ambiguity may lie in a single word, a phrase, or even a whole sentence. Furthermore, ambiguities may manifest as the result of an inadequate form of expression as well as from incomplete information [4]. Requirements specifications are generally described in long, complex sentences, thus increasing the probability that they present some sort of ambiguity [14]. Also, the context in which specifications are produced usually differs from their context of interpretation, giving raise to further challenges [29].

In view of the need to simplify requirement descriptions while also preventing the problems of decontextualization, we advance the notion of 'kernel sentences' as a useful point of departure to specify requirements. A kernel sentence (KS) is defined as a simple, declarative sentence with only one verb, all the arguments of which are overt and explicit.

KSs are simple and self-contained units which can help reduce some of the ambiguities found in requirements specifications [7]. In order to solve other issues, such as duplication and inconsistency, Zhao et al. [38] recommend that the requirements be summarised and organised using a model synthesis technique. The synthesis of a large set of requirements into a conceptual model not only improves their quality but is an interesting technique for coping with the complexity of the domain [11].

This paper proposes a method that receives a natural language specification as input, inspects it for compliance with KS structure, and summarises the specification by deriving from the input text a conceptual model that includes entities and relationships. Our method is intended to help the requirements engineer achieve better quality specifications, and can be applied under various circumstances. For instance, if employed by an individual analyst who elicits requirements from different people, the method could

help achieve a consolidated summary of the whole landscape, providing the analyst with a consistent and coherent specification. More significantly, the method could be used by a group of analysts working together, so the resulting conceptual model ensures the consistency of requirements. Finally, if an organisation already has a large amount of consolidated documentation that needs to be better organised, the method could help in this challenge.

This is a revised version of the method presented in Antonelli et al. [1]. Improvements on the identification of multi-word entities and different types of relationships are described. This paper also describes a prototype that was developed to support the proposed method. This prototype manages different types of requirements specifications in natural language (ranging from plain text to complex Use Cases), checks KS structure, and renders a conceptual model graphically. Finally, the paper presents some preliminary evaluations of the method presented in [1] using SUS survey [8][9] to measure its applicability. We assessed (i) the recommendation to improve specifications by applying KSs, and (ii) the summary resulting from the conceptual model. The prototype tool was not assessed.

The rest of the paper is organised as follows: Section 2 defines kernel sentences. Section 3 details our contribution, namely the method proposed. Section 4 describes the tool that supports the method. Section 5 presents the preliminary evaluation. Section 6 is a review of some related work. Finally, Section 7 discusses our conclusions.

2 Kernel Sentences

Kernel sentences (sometimes referred to as 'basic sentences') are those with a minimal number of obligatory grammatical attributes. This means that a kernel sentence fulfils the following basic conditions [26]:

(1) It has only one verbal head.
(2) It is unmarked in mood, therefore it is declarative.
(3) It is unmarked in voice, therefore it is in the active voice.
(4) It is unmarked in polarity, therefore it is affirmative.

This theoretical construct was first introduced in 1957 by Z. S. Harris [18] and featured in the early work of Noam Chomsky [10]. Broadly, these grammarians have argued that the more complex (non-kernel) structures are built upon a number of simpler underlying elements (the kernel forms). Thus, a sentence such as "The ripe tomatoes are harvested by a farmer" can be decomposed into the following kernels: "The tomatoes are ripe," and "A farmer harvests the tomatoes."

Alternatively, a KS could be understood as a primitive construction which has not undergone any transforms. This would also exclude the processes of deictic reference and ellipsis. Deixis is the use of a word such as a pronoun or adverb to refer to something in the context instead of naming it. For instance, in "He buys supplies there," the pronoun "he" points to a male third person in the singular, whereas "there" is a location far from the speaker—*who* this person is, and *where* the supplies are actually bought, must be inferred from context. Ellipsis, on the other hand, is the omission of words that are recoverable from the context. In other words, the name of an entity is substituted

with an empty element. Thus, in the sentence "The farmer fertilizes in the summer," the direct object of "fertilize" has been elided and, again, *what* is fertilized must be inferred from context. The last two examples are not kernel sentences, whereas the following are: "The stock manager buys supplies at ABC depot," "The farmer fertilizes the soil for tomatoes in the summer." Put in formal terms, kernel sentences should also meet the following conditions:

(5) All verbal arguments are overt.

(6) All verbal arguments have explicit (i.e., non-deictic) referents.

In sum, kernel sentences are defined by conditions (1)-(6). This notion will serve as a useful point of departure for improving requirements specifications by reducing sources of ambiguity.

3 Our Contribution

We devised a method for deriving a rich conceptual model from a set of specifications described in natural language. This method makes it possible to (i) extract relevant entities (such as actors and resources), (ii) identify the behaviours associated with each entity, and (iii) establish various kinds of relationship among entities.

Our approach consists of two steps: first, the input (a set of specifications described in natural language) is checked for compliance with KS format; and second, a conceptual model is derived by applying a set of heuristic rules.

This approach can be applied to different kinds of artifacts, ranging from plain text specifications to more structured artifacts such as User Stories or Use Cases. It is important that the artifacts contain sentences in kernel format. Therefore, the first step of our approach consists in checking sentence by sentence to ensure that kernel format is used throughout. At this stage, the analyst should look for sentences containing tacit subjects, multiple verbs, verbs in the passive voice, and any other feature that is not compatible with kernel format. The sentences are rewritten as necessary. It may be necessary to consult with the experts to fill in any missing information or to resolve ambiguities.

Once kernel sentence verification is completed, a conceptual model can be derived. In our approach, this is done by analysing the syntactic dependencies established by the verb. First, the main verb of the sentence is identified and classified. Depending on the type of verb, we can distinguish relationships (which relate two entities) and behaviours (which apply to single entities). Next, we look for the relevant entities affected by the verb. In other words, it is necessary to identify the main verbal arguments—that is, subject and objects [3]. Subjects and objects are expressed as noun phrases (NP), which can be structurally complex. In our approach, attention is directed first to the nominal heads directly dependent on the verb. Multi-word entities are then extracted by exploiting the NP. Finally, the appropriate relationships/behaviour is assigned to the entities thus identified. Figure 1 summarizes our approach.

Let's consider an example. The following specification, although grammatically correct, does not satisfy KS format: "The client of the bank opens and closes an account. The client deposits and withdraws in the account." The analysis performed in the first

step of our approach will reveal that there are multiple verbs within the sentences. This feedback will require that the analyst confirm whether the four actions (opening, closing, depositing, and withdrawing) can all be performed by the client. The sentences are rewritten accordingly.

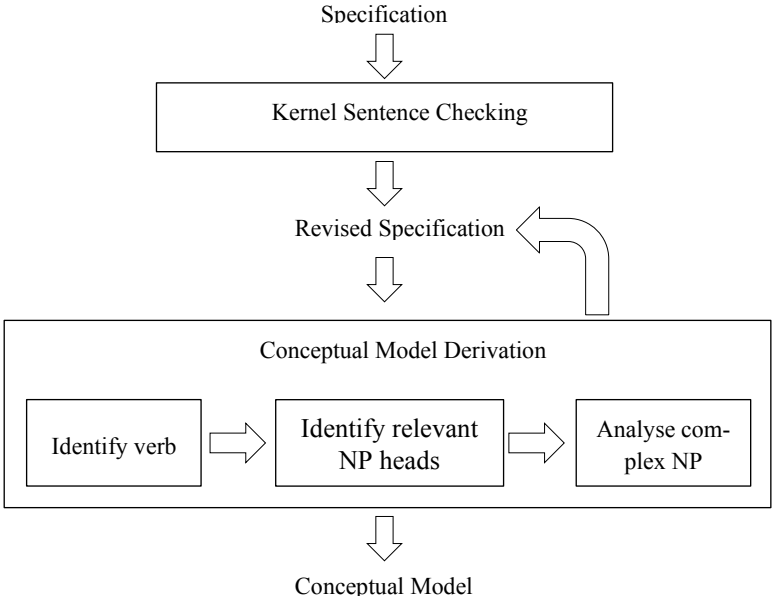

Fig. 1. Synthesis of our approach.

Now let's consider how a conceptual model is derived from the revised KS "The client of the bank opens an account." The main verb is "open," which is classified as a transitive verb. Following our heuristics (described in 3.1), this means that a non-hierarchical relationship is established between two entities, and that one of the entities possesses the behaviour "to open". There are three nouns in this sentence: "client," "bank," and "account." However, only two of them depend directly on the verb, namely "client" (which is the subject head) and "account" (the direct object head).

So far in our example we identified a non-hierarchical relationship between the simple entities "client" and "account." Next, we verify whether these entities can be extended by exploring the noun phrases of which they are heads. Applying further rules, we can finally recognize a complex entity "client of the bank." Figure 2 summarizes this example.

Continuing our example, let's consider the revised sentence "The client deposits in the account." Again, the analysis begins by identifying the verb "deposit," which also establishes a non-hierarchical relationship between two entities. However, there is only one NP head directly related to the verb, namely "client." Analysis of syntactic dependencies reveals that a direct object is missing: the client deposits *what* in the account? (Here "account" belongs to the prepositional phrase headed by "in"; normally a

prepositional phrase cannot be a direct object.) This example shows how conceptual model derivation may reveal omissions, inconsistencies and other kinds of errors in the requirements, implying that the sentence needs further revision. Thus the upward arrow in Figure 1.

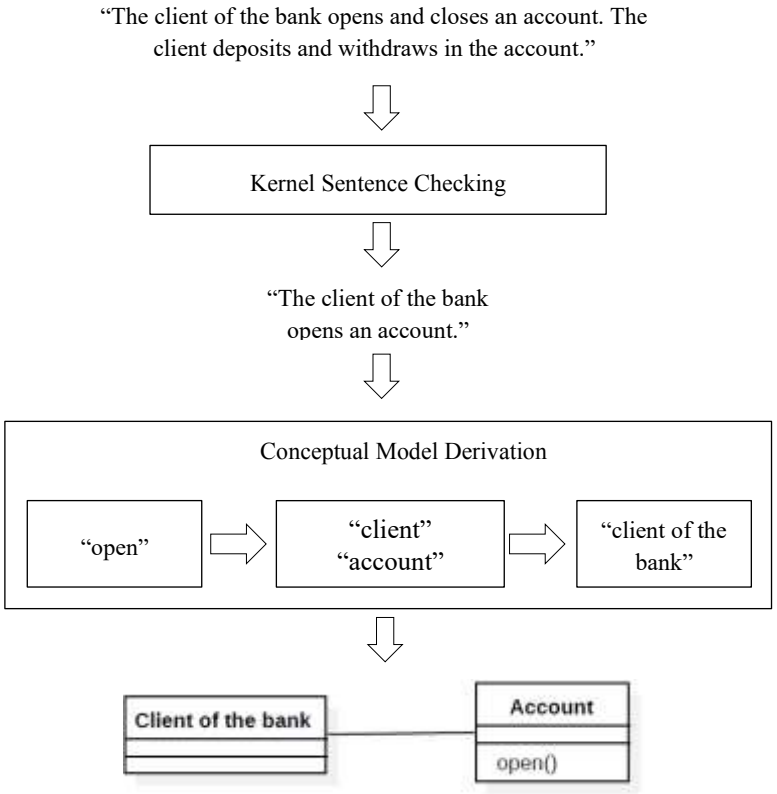

Fig. 2. An example of conceptual model derivation.

The following subsections describe the heuristic rules that are applied to derive a conceptual model.

3.1 Identifying Relationships and Behaviour

Our method begins by identifying a verb and classifying it in order to determine which kind of relationship it establishes, how many entities are affected, and which entity actually realizes the behaviour.

We distinguish three types of relationship:

(i) hierarchical relationships (also known as "is a" relationships)

(ii) non-hierarchical relationships (also known as "knowledge" relationships)

(iii) possessive relationships (which include both aggregation and composition relationships)

We also classified verbs into four groups, each of which is associated to a particular type of relationships. The four groups are: (i) copulative verbs; (ii) possessive verbs; (iii) transitive verbs; and (iv) intransitive verbs.

First, copulative verbs (such as "to be") establish a hierarchical relationship between two entities and no behaviour. The subject of a copulative sentence is the subclass, whereas the superclass is either the nominal attribute or a concatenation of the nominal subject and the adjectival attribute. Figure 3 shows an example of each case.

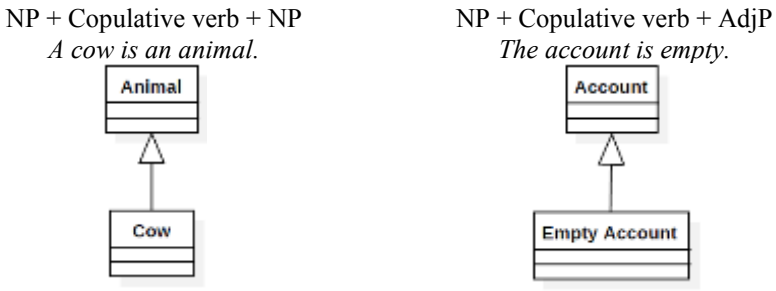

Fig. 3. Hierarchical relationships derived from copulative structures.

A second group consists of verbs with a possessive meaning, such as "have," "include," or "contain." These verbs establish a possessive (part-whole) relationship between two entities and no behaviour. The subject is the 'possessor', and the direct object is the 'possessed.' We also include in this group the existential structure "there is/there are". In this case, the 'possessor' is contained in the locative prepositional phrase and the 'possessed' is the attribute of the verb "be." Figure 4 shows examples of possessive sentences.

Fig. 4. Possessive relationships.

The third group is that of transitive verbs, that is, verbs which take a subject and at least one object, such as "buy," "activate," or "send." These verbs establish a non-hierarchical relationship between the subject and object entities, and indicate a behaviour of the direct object entity. Figure 2 above contains an example of transitive verb "open."

Finally, the fourth group includes two types of verbs: (a) intransitive verbs, that is, verbs which do not take any objects, such as "wait," "bloom," or "shake;" and (b) reflexive verbs, or verbs where the subject and object are the same entity, as in "The door locks automatically." These verbs do not establish any relationship but only indicate a behaviour of the subject entity. Figure 5 shows examples with verbs in this group.

Many verbs can be either transitive or intransitive, sometimes with a difference in meaning. The analyst must not mistake a transitive structure with an elided object (as in “The client deposits in the account,” see above) for an intransitive structure.

(a) *The tomato fruit ripens after 30 days.* (b) *The door locks automatically.*

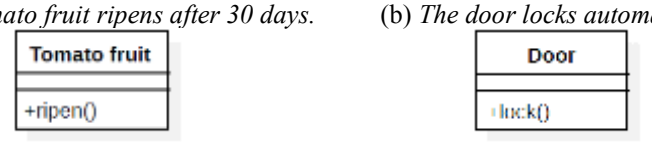

Fig. 5. An intransitive verb (a) and a reflexive verb (b).

3.2 Extracting Multi-Word Entities

Once the heads of the subject and object NPs have been identified, multi-word entities can be extracted by finding the syntactic dependents of these heads. Related research in the area of terminology extraction have shown that specific NP patterns are language- and domain-dependent (e.g. [33][13]).

We suggest looking for the longest possible NP, ignoring purely functional elements such as determiners “the” and “a”. Since our approach is based on kernel sentences, which by definition have only one verb, the complexity of NPs is reduced significantly. Consequently, multi-word entities will generally be expressed by means of a noun combined with other nouns, adjectives, and/or prepositional phrases. Also, by searching first for the subject and object heads, which always have associated relations or behaviours, the burdensome task of filtering through empty candidate entities is eased.

A multi-word entity could be decomposed into further entities, with associated relations and behaviours. Such possibility, however, will not be considered in this article.

4 Supporting Tool

A software tool was implemented to support the method proposed in this article. The prototype is a web application with a service-oriented architecture. The core of the application and its services are implemented in Python [32], whereas the web components use Django [12], and the APIs use Flask [15]. Python is also used to communicate with natural language processing libraries provided by Spacy [36] and NLTK [27]. The application is responsive; users can therefore contribute to the acquisition of knowledge from a variety of platforms such as desktop computers and mobile phones.

The prototype implements two user roles: (i) project administrators, and (ii) users (experts and analysts). Project administrators create a project, add users, and define the type of artifact to be used for the specifications. The artifacts are based on natural language. The prototype provides the means to define a structure for the artifacts. For example, instead of using plain sentences the structure could be defined as User Stories with three attributes (“As a,” “I want,” and “so as to”) or as Use Cases with more attributes (id, actor, goal, happy path, alternative path, exceptional path, and so on).

Experts and analysts can contribute by adding new artifacts or editing someone else’s artifacts. While the user types in the specification text, the prototype checks for

compliance with kernel format. If a conflict is detected, the cause of the error is informed to the user. Figure 6 is a screenshot showing that non-kernel lack of subject was detected.

Fig. 6. The prototype detected a non-kernel sentence.

Among other features, the prototype implements a module that derives a UML class diagram from a set of natural language sentences. The module consists of five main subprocess that are run in the following order: (i) a subprocess identifies and classifies the sentence root; (ii) a subprocess identifies the subject and object heads, and creates a preliminary relationship record depending on the type of root; (iii) a subprocess creates a preliminary behaviour record, depending on the type of root; (iv) a subprocess extracts multi-word entities by matching patterns that contain the subject and object heads and updates the relationship and behaviour records; and finally (v) a subprocess merges all the relationship and behaviour records into individual class records containing entity name, relationships (including relationship type and target entity), and associated behaviours. This information is finally translated into a UML class diagram using the PlantUML [30] framework.

The module uses Spacy's dependency matcher [37] to process the syntactic relations among sentence tokens. In contrast with traditional matchers, the dependency matcher allows us to analyse patterns independently of the position of tokens in the sentence. Dependency matchers are particularly useful when analysing languages with a flexible word-order such as Spanish.

5 Preliminary Evaluation

This section describes an evaluation of the approach proposed in [1], upon which the current article is based. The former approach also describes a method to derive simple conceptual models from kernel sentences. The results of this evaluation will serve as a point of comparison for the new approach proposed here (evaluation in progress).

We assessed the applicability of the processes of kernel sentence checking and conceptual model derivation. The evaluation was applied to the domain of agriculture using scenarios [21]. In particular, a set of existing scenarios describing the process of tomato production was used. The content of the scenarios ranged from basic agricultural concepts to advanced concepts and techniques. The participants to the evaluation were 14 students of a post-graduate course on requirements engineering. All participants had experience in the industry of software development; however, none of them had

experience as farmers (except for some basic experience related to agriculture, mainly gardening). Thus, the participants played the role of requirements engineers who obtained information from domain experts (farmers) and needed to consolidate the knowledge gathered through scenarios.

We provided the participants with the scenarios and asked them to check for compliance with KS format and to rewrite the scenarios when necessary. Since KS rules are based on intuitive linguistic competence, the participants were able bring the scenarios to KS format with ease. Once the scenarios had been corrected, we asked the participants to apply the rules proposed in [1] to derive a conceptual model. The participants worked on artifacts they had not written and were able to improve them. The participants were also able to derive a conceptual model from the artifacts. After the evaluation, the participants showed a good understanding of the knowledge described in the scenarios.

The Systems Usability Scale (SUS) [8][9] was used to evaluate the results of the case study regarding the applicability of the former approach. Although SUS is mainly applied to the assessment of software systems usability, it has proved to be an effective means to assess products and processes [2]. SUS consists of a ten-item questionnaire. Each question must be marked on a scale ranging from 1 (“Strongly disagree”) to 5 (“Strongly agree”). The ten items are grouped in pairs asking the same question from a complementary point of view in order to obtain high confidence results. Calculation of the SUS score is performed as follows: first, items 1, 3, 5, 7, and 9 are scored considering the value ranked minus 1; then items 2, 4, 6, 8, and 10 are scored considering 5 minus the value ranked. Next, each participant’s scores are added up and multiplied by 2.5 to obtain a new value ranging 0-100. Finally, the average is calculated. There are three possible outcomes: “Non acceptable” 0-64, “Acceptable” 65-84, and “Excellent” 85-100 [25]. The score obtained in our evaluation was 69.73; thus the approach is considered “Acceptable.”

6 Related Work

This is a revised version of the method presented in Antonelli et al. [1]. We take from their work the concept of KS and adjust it by adding two more conditions to the definition. Also, instead of separating the identification of entities and relationships as two different processes, we reorganised the algorithm so that entity extraction relies on relationship identification. This logical reorganisation of both processes, which distinguishes our contribution from most other related works, enables us to reduce the number of meaningless entities in the derived conceptual model.

Shuttleworth et al. [35] propose a semi-automatic approach to generate a conceptual model from descriptions of a phenomenon. Narratives describing the problem are transformed into a list of concepts and relationships and visualized using a network graph. They use pattern-based grammatical rules and an NLP dependency parser as we do, but they do not use machine learning techniques. Moreover, they are concerned about simulation process.

Robeer et al. [34] and Lucassen et al. [24] propose a method to derive conceptual models automatically from User Stories. Their main aim, however, is to identify dependencies, redundancies, and conflicts. Also, they advocate for a fully automated approach with no human participation, whereas we believe that human participation is vital in the interactive construction of knowledge. Gupta et al. [17] also are interested in User Stories and attempt to solve the problem of ambiguity through the derivation of a conceptual model. Their approach is iterative and incremental, providing information while the User Stories are described.

Fliedl et al. [16] describe the work performed within the NIBA project and present a strategy to analyse complex sentences such as 'if/then-constructions' and to transform them into dynamic model components. Although this is not a classical conceptual model, their analysis of conditional sentences is interesting. More recently, Kop et al. [20] focus on semi-automatic derivation. They claim that there is no direct correspondence between natural language elements (e.g., words) and conceptual model elements, and therefore require the transformation into an interlingua, wherein the designer plays an important part.

Letsholo et al. [22] propose a tool for automatically constructing analysis models from natural language in the context of model driven development. Their approach relies on a set of conceptual patterns. Although interesting, the results of their approach are conditioned by the patterns used. Kashmira et al. [19] concentrate on the derivation of an entity relationship model from natural language specifications. Thus they only deal with concepts (entities and attributes), relationships, and constraints, whereas our approach is concerned with entities, relationships, and behaviours. Bogatyrev et al. [6] present a framework for conceptual modelling and advance modelling techniques that combine the usage of conceptual graphs and Formal Concept Analysis. Conceptual graphs serve as semantic models of text sentences and the data source for formal context of concept lattice. We believe that incorporating semantic analysis can improve the analysis in general domains; however, specific application domains usually have specific semantics that cannot be captured by general tools.

7 Conclusions and future work

This paper presents an approach to derive a conceptual model from specifications written in natural language. The proposed approach also provides guidance on the writing of specifications, suggesting the adoption of kernel format to reduce ambiguity. We also describe a software prototype that uses natural language processing and artificial intelligence tools to support the method. Further work is needed to continue improving the prototype. For instance, it would be interesting to evaluate LLMs for such processes as KS compliance checking. Our main concern, however, is the improvement of the method, and not to develop a full application.

The approach can be used in different circumstances: to summarize documents produced by other people, to consolidate and summarize specifications elicited by a single analyst from multiple sources, or to consolidate and summarize requirements produced collaboratively by a group of analysts or experts. In either case, the objective of the

approach is to provide an overview of the consolidated knowledge in order to obtain consistent and coherent specifications.

Our contribution is part of a greater enterprise. Our aims are not restricted to a particular type of artifacts but rather consider different inputs, including large documents produced for legacy systems, short pieces of information such as instant messaging produces, and specifications produced by experts. We have already developed and tested some pieces of the general approach and continue working in the development of the tool, mainly improving its usability and correcting performance issues. Usability is essential to make users adopt the tool, and hence the approach. Performance concerns are also important insofar as different strategies to check conditions and to derive new information are time consuming. New evaluations of the revised method remain to be done.

Acknowledgments. This paper is partially supported by funding provided by the STIC AmSud program, Project 22STIC-01.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Antonelli, L., Delle Ville, J., Adorno, M. A., Ballestero, L. P., Cecconato, S. A., Fernández, A., Maclen, G., Maltempo, G., Mattei, J. E., Tanevich, L., Torres, D.: An Approach to Extract a Conceptual Model from Natural Language Specifications. In: Workshop in Requirements Engineering (WER), Porto Alegre, Brazil (2023)
2. Bangor, A., Kortum, P. T., Miller, J. T.: An empirical evaluation of the system usability scale. Intl. Journal of Human–Computer Interaction 24(6), 574-594 (2008)
3. Bashir, N., Bilal, M., Liaqat, M., Marjani, M., Malik, N., Ali, M.: Modeling Class Diagram using NLP in Object-Oriented Designing. In: 2021 National Computing Colleges Conference (NCCC), pp. 1-6, Taif, Saudi Arabia (2021)
4. Berry, D., Kamsties, E., Krieger M.: From Contract Drafting to Software Specification: Linguistic Sources of Ambiguity. Handbook, University of Waterloo (2003)
5. Boehm, B.W.: Software Engineering, Computer society Press, IEEE (1997)
6. Bogatyrev, M., Samodurov, K.: Framework for Conceptual Modeling on Natural Language Texts. CDUD@CLA, 13-24 (2016)
7. Boyd, N. S.: Using Natural Language in Software Development. Journal of Object-Oriented Programming - Report on Object Analysis and Design, 11-9 (1999)
8. Brooke, J.: SUS-A quick and dirty usability scale. In: Jordan, P. W., Thomas, B., McClelland, I. L., Weerdmeester, B. (eds.), Usability evaluation in industry, pp. 189-194 (1996)
9. Brooke, J: SUS: a retrospective. Journal of usability studies 8(2), 29-40 (2013)
10. Chomsky, N.: The Logical Structure of Linguistic Theory. Plenum Press, New York (1975)
11. Dick, J., Hull, E., Jackson, K.: Requirements Engineering, 4th edition, Springer (2011)
12. Django Homepage, https://www.djangoproject.com/, last accessed 2024/03/15
13. Fang, A. C., Cao, J., Song, Y.: A New Corpus Resource for Studies in the Syntactic Characteristics of Terminologies in Contemporary English. In: TIA (2009)

14. Ferrari, A., Spagnolo, G. O., Gnesi, S.: PURE: A Dataset of Public Requirements Documents. In: 25th International Requirements Engineering Conference (RE), pp. 502-505, Lisbon, Portugal (2017)
15. Flask Documentation, https://flask.palletsprojects.com/, last accessed 2024/03/15
16. Fliedl, G., Mayerthaler, W., Winkler, C., Kop, C., Mayr, H. C.: Enhancing requirements engineering by natural language based conceptual predesign. In: IEEE International Conference on Systems, Man, and Cybernetics (Cat. No.99CH37028), pp. 778-783, Tokyo, Japan (1999)
17. Gupta, A., Poels, G., Bera, P.: Creation of Multiple Conceptual Models from User Stories – A Natural Language Processing Approach. In: Advances in Conceptual Modeling. ER 2019. Lecture Notes in Computer Science, vol 11787. Springer (2019)
18. Harris, Z. S.: Co-Occurrence and Transformation in Linguistic Structure. Language 33(3-1) 283-340 (1957)
19. Kashmira, P. G. T. H., Sumathipala, S.: Generating Entity Relationship Diagram from Requirement Specification based on NLP. In: 3rd International Conference on Information Technology Research (ICITR), pp. 1-4, Moratuwa, Sri Lanka (2018)
20. Kop, C., Fliedl, G., Mayr, H.: From Natural Language Requirements to a Conceptual Model. In: International Workshop on Design, Evaluation and Refinement of Intelligent Systems (DERIS2010), pp 646-67 (2010)
21. Leite, J. C. S. d. P., Rossi, G., Balaguer, F., Maiorana, V., Kaplan, G., Hadad, G., Oliveros, A.: Enhancing a requirements baseline with scenarios. Requirements Engineering 2(4), 184-198 (1997)
22. Letsholo, K. J., Zhao, L., Chioasca, E. V.: TRAM: A tool for transforming textual requirements into analysis models. In: 28th IEEE/ACM International Conference on Automated Software Engineering (ASE), pp. 738-741, Silicon Valley, CA, USA (2013)
23. Lim, S. L., Finkelstein, A.: StakeRare: Using Social Networks and Collaborative Filtering for Large-Scale Requirements Elicitation. IEEE transactions on software engineering 38(3) 707-735 (2012)
24. Lucassen, G., Robeer, M., Dalpiaz: Extracting conceptual models from user stories with Visual Narrator. Requirements Engineering 22, 339–358 (2017)
25. McLellan, S., Muddimer, A., Peres, S. C.: The effect of experience on System Usability Scale Ratings. Journal of Usability Studies 7, 56-67 (2012)
26. Moitra, S.: Generative Grammar and Logical Form. In: Pranab, K. S. (ed.), Logic Identity and Consistency. Allied Publishers (1998)
27. NLTK Documentation, https://www.nltk.org/, last accessed 2024/03/15
28. Norlyk, B.: Miscommunication and discourse practices in occupational cultures. International Journal of Applied Linguistics 6(1), 7-20 (1996)
29. Nystrand, M.: The Role of Context in Written Communication. The Nottingham Linguistic Circular 12(1), 55-65 (1983)
30. PlantUML Homepage, https://plantuml.com/, last accessed 2024/03/15
31. Potts, C.: Using schematic scenarios to understand user needs. In: Proceedings of the 1st conference on Designing interactive systems: processes, practices, methods, & techniques (1995)
32. Python Homepage, https://www.python.org/, last accessed 2024/03/15
33. Rico, M., Calleja, P., Martin, P., and Montiel, E.: Extracting terminologies in the legal domain: a syntactic pattern-based approach for Spanish. In: Iberlegal workshop at JURIX conference (2019)

34. Robeer, M., Lucassen, G., van der Werf, J. M. E. M., Dalpiaz, F., Brinkkemper, S.: Automated Extraction of Conceptual Models from User Stories via NLP. In: 24th International Requirements Engineering Conference (RE), pp. 196-205, Beijing, China (2016)
35. Shuttleworth, D. Padilla, J.: From Narratives to Conceptual Models via Natural Language Processing. In: 2022 Winter Simulation Conference (WSC), pp. 2222-2233, Singapore (2022)
36. SpaCy Homepage, https://spacy.io/, last accessed 2024/03/15
37. SpaCy Dependency Matcher, https://spacy.io/api/dependencymatcher, last accessed 2024/03/15
38. Zhao, L., Alhoshan, W., Ferrari, A., Letsholo, K. J., Ajagbe, M. A., Chioasca, E., Batista-Navarro, R. T.: Natural Language Processing for Requirements Engineering: A Systematic Mapping Study. ACM Computing. Surveys 54(3) 1-41 (2021)