Treatment of Passive Voice and Conjunctions in Use Case Documents

Leonid Kof

Fakultät für Informatik, Technische Universität München,
Boltzmannstr. 3, D-85748 Garching bei München, Germany
kof@informatik.tu-muenchen.de

Abstract. Requirements engineering, the first phase of any software development project, is the Achilles' heel of the whole development process, as requirements documents are often inconsistent and incomplete. In industrial requirements documents natural language is the main presentation means. In such documents the system behavior is specified in the form of use cases and their scenarios, written as a sequence of sentences in natural language. For the authors of requirements documents some facts are so obvious that they forget to mention them. This surely causes problems for the requirements analyst.
Missing information manifests itself, for example, in sentences in passive voice: such sentences just say that some action is performed, but they do not say who performs the action. In the case of requirement analysis this poses a serious problem, as in every real system there is an actor for every performed action.
There already exists an approach able to guess missing actors and actions. However, the existing approach is able to handle sentences containing exactly one verb only. The approach presented in this paper extends the existing one by treatment of compound sentences and passive voice. Feasibility of the presented approach to the treatment of passive and conjunctions was confirmed in a case study.

1 Document Authors are not Aware that some Information is Missing

Some kind of requirements document is usually written at the beginning of every software project. The majority of these documents are written in natural language, as the survey by Mich et al. shows [1]. This results in the fact that the requirements documents are imprecise, incomplete, and inconsistent. The authors of requirements documents are not always aware of these document defects. From the linguistic point of view, document authors introduce three defect types, without perceiving them as defects (cf. Rupp [2]): [1]

Deletion: "…is the process of selective focusing of our attention on some dimensions of our experiences whereas excluding other dimensions. Deletion reduces the world to the extent that we can handle."

Generalization: "…is the process of detachment of the elements of the personal model from the original experience and the transfer of the original exemplary experience to the whole category of objects."

[1] The following definitions are translations of the definition from [2] (in German)

Distortion: "…is the process of reorganization of our sensory experience."

It is one of the goals of requirements analysis, to find and to correct the defects of requirements documents.

In requirements documents the behavior of the prospective system is often specified as a set of *use cases*, each use case represented by one or several *scenarios* (cf. Rupp [2]). A scenario is a sequence of natural language sentences. Each sentence of this sequence represents either some input to the system or the reaction of the system to previous inputs.

The presented paper focuses on the "deletion"-defects in scenarios. Deletion manifests itself in scenarios in the form of missing action subjects or objects or even in whole missing actions. One of the reasons for the deletion may be the fact that some information is too obvious for the author of the requirements document, so that she finds it unnecessary to write down this information. One further reason for missing action subjects, manifesting itself in sentences in passive voice, can be the absence of an exact construction plan, typical in the early stages of the project. It is the goal of the approach presented in this paper, to identify missing parts of scenarios written in natural language and to produce message sequence charts (MSCs) containing the reconstructed information. (See Section 3 for an introduction to MSCs.)

For the remainder of the paper we use the following terminology: A *scenario* is a sequence of natural language sentences, each sentence representing some *action*. A *message sequence chart (MSC)* is a set of *communicating objects* and a sequence of *messages* sent/received by these objects.

The remainder of the paper is organized as follows: Section 2 introduces the case study used to evaluate the presented approach. Section 3 introduces message sequence charts (MSCs) and an existing approach transforming scenarios to MSCs. This approach works only for sentences in active voice, containing exactly one verb. Section 4 explains an extention of this approach, allowing both for passive voice and for several verbs in the same sentence. Section 5 presents the evaluation of the approach on a case study. Finally, Sections 6 and 7 present an overview of related work and the summary of the paper, respectively.

2 Case Study: The Instrument Cluster

Authors of requirements documents tend to forget to write down facts that seem obvious to them. Even in a relatively precise requirements document, as for example the instrument cluster specification [3], some missing facts can be identified. The instrument cluster specification describes the optical design of one part of the car dashboard (the instrument cluster), its hardware, and, most importantly, its behavior. The behavior is specified as a set of scenarios, like this:

1. The driver switches on the car (ignition key in position ignition on).
2. The instrument cluster is turned on and stays active.
3. After the trip the driver switches off the ignition.
4. The instrument cluster stays active for 30 seconds and then turns itself off.
5. The driver leaves the car.

There are apparent problems if we try to translate this scenario to a sequence of messages exchanged by communicating objects. Firstly, there is no one-to-one correspondence between sentences and messages. For example, sentences number 2 and 4 contain two potential messages each: Sentence 2 contains actions “The instrument cluster is turned on” and “The instrument cluster stays active” and sentence 4 contains actions “The instrument cluster stays active for 30 seconds” and “The instrument cluster turns itself off”. Furthermore, for at least one of these actions (“The instrument cluster is turned on”) the actor is not explicitly specified. It is the goal of the approach presented in this paper, to resolve such incomplete specifications and present the results to a human analyst for validation.

3 Scenarios and Message Sequence Charts

Message Sequence Charts (MSCs) are a convenient means for concise and precise representation of action sequences. An MSC consists of a set of communicating objects. These communicating objects exchange messages, whereas every message has a well defined sender and receiver. Graphically, communicating objects are represented as rectangles, and messages as arrows; the time line is directed top down (cf. Figure 1).

When translating scenarios (written in natural language) to MSCs, it is necessary to deal with typical deficiencies of natural language texts: It can happen that either the message sender or the receiver are not explicitly mentioned, or the whole action is just omitted. For example, if we directly translate the scenario introduced in Section 2 to an MSC, a possible translation is the MSC in Figure 1[2]. The problems of this translation are apparent: there are definitely missing messages from the car to the instrument cluster, otherwise the instrument cluster cannot know that it should be turned on or off. Furthermore, some sentences, like “The instrument cluster is turned on”, do not specify the message receiver. Even if we rephrase this sentence to active voice (“The instrument cluster turns on”), the message receiver remains unspecified.

The problem of unspecified message senders/receivers and missing actions was solved in [4] by the organization of MSC messages in a stack. Organization of messages in a stack is motivated by the idea of situation stack by Grosz et al. [5]. Grosz et al. introduce a situation stack to explain how the human attention focuses on different objects during a discourse. The focus depends on the sequence of sentence heard so far. By default, a sentence defines some situation and is pushed onto the stack. If a sentence reverts the effect of some previous sentence, the corresponding stack element is popped:

John enters the shop	//push “enter”
— Some actions in the shop —	
John leaves the shop	//pop “enter” and the above stack elements

The idea of the situation stack can be easily transferred to MSCs: It is possible to define an active object as an object that has sent a message but has not received an answer yet. If the receiver of the message under analysis (msg) is an active object, then it is possible to find the topmost message of the stack sent by this object (msg'). Then,

[2] To make the figure compacter, “instrument cluster” is abbreviated as “ins. clust.”.

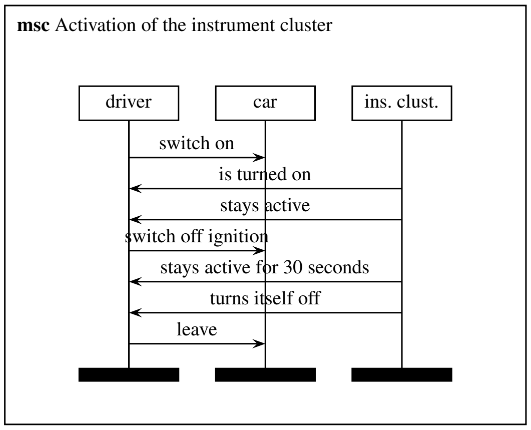


Fig. 1. Scenario "Activation of the instrument cluster", manual translation to MSC

msg' and the messages contained in the stack above it are popped. If the receiver is not an active object, the message under analysis is pushed onto the stack.

The organization of messages in a stack makes also the identification of missing messages possible: If the sender of the message under analysis ($sender_{new}$) differs from the receiver of the message on the top of the stack (rec_{top}), then the message from rec_{top} to $sender_{new}$ is missing. For example, for the MSC in Figure 1, missing messages from "car" to "instrument cluster" just after messages "switch on" and "switch off ignition" can be identified in this way. The message stack enables the identification of missing message senders and receivers as well: The default message sender/receiver equals to the receiver/sender of the message on the top of the stack. The details of the above algorithm can be found in [4].

The procedure of identification of senders and receivers apart from the analysis of the stack, as implemented in [4], is rather simple: It is assumed that every sentence contains exactly one verb. Furthermore, it is assumed that there exists a previously constructed list of potential communicating objects (glossary). Then, the longest word sequence before/after the verb that is contained in the glossary is identified as the message sender/receiver. If no such word sequence is found in the sentence, the sender/receiver remains unspecified for the concrete sentence. In this case the sender/receiver is augmented by the means of stack analysis.

The requirement that every sentence contain exactly one verb is obviously violated in passive and compound sentences. For example, the sentence "The instrument cluster is turned on and stays active" contains three verbs: "is", "turned", "stays". In the case study performed in [4] such sentences were manually split and rewritten, so that in the resulting text every sentence contained exactly one verb. It is the goal of the approach presented in this paper, to extend the procedure of the identification of senders and receivers implemented in [4] onto sentences containing several verbs. This includes both passive voice and compound sentences.

4 Compound Sentences and Passive Voice: Translation to MSC Messages

The basic idea for the translation of compound sentences to MSC messages is fairly simple: We split every sentence into *elementary segments* and translate every segment to an MSC message. An *elementary segment* is defined as a sentence segment that does not contain any conjunctions or commas/colons. For example, the translation of the sentence “The instrument cluster is turned on and stays active” consists of two messages: Some unspecified object sends the command “turn on” to the instrument cluster and receives the answer “stays active” from the instrument cluster.

Generally, we want to take following issues into account when translating sentences to MSC messages:

- If we split the original sentence into elementary segments, it can happen that one of the segments lack the grammatical subject. For example, the sentence “The instrument cluster is turned on and stays active” would be split into “the instrument cluster is turned on” and “stays active”. The second segment lacks the subject. However, the subject is necessary to identify the message sender. This problem can be solved by propagation of the grammatical subject from the first segment of the sentence to the second one.
- Even when the grammatical subjects of the sentence segments coincide, the senders of the corresponding messages can differ. This is due to the fact that in passive sentences the grammatical subject corresponds to the message receiver, not to the sender. For example, in the translation of the segment “the instrument cluster is turned on”, “instrument cluster” is the receiver of the message “turn on”.
- If the sentence consists of several parts and some parts do not contain an own verb, the verbs should be accordingly propagated. For example, in the sentence “The driver drives more than 30 km/h and less than 50 km/h” the verb “drives” should be propagated to the segment “less than 50 km/h”.

As the technical means for splitting the sentences into segments and for identification of the verb we use a part-of-speech (POS) tagger in the presented approach. Such a tagger assigns a POS-tag (substantive, verb, adjective, . . .) to every word. Currently available taggers, as for example the tagger by Ratnaparkhi [6], have the precision of about 97%, which makes them unlikely to become an extra error source. The translation of tagged sentences to MSC messages goes in five steps:

1. The tagged sentences are split into elementary segments, not containing any conjunctions or commas/colons.
2. Every sentence segment is annotated as either active or passive or sentence segment without any verb.
3. For every sentence segment, the grammatical subjects, objects, and verbs are extracted, if possible.
4. The extracted grammatical subjects and objects are propagated to other sentence segments, if necessary.

5. Finally, for active segments the subjects are declared to message senders and objects to message receivers. For passive segments the assignment of senders and receivers is the opposite.

Every of these steps is explained below in detail.

Splitting of tagged sentences: The POS tagger by Ratnaparkhi appends a tag to every word using the underscore, so that the tagged sentence looks like this:

> The_DT instrument_NN cluster_NN is_VBZ turned_VBN on_RP and_CC stays_NNS active_JJ ._.

This form allows to split every sentence into elementary segments. As splitting marks we use the regular expression matching conjunctions: `" [^ ]*_CC "` (space, followed by a character sequence without spaces, followed by underscore, followed by the conjuction tag, followed by space)[3], and also regular expressions matching tagged punctuation: `" [^ ]*_, "` (matching coma), `" [^ ]*_\.[ ]*"` (matching period), and `" [^ ]*_:[ ]*"` (matching colon). The splitting mark matching conjunctions splits the sentence "The instrument cluster is turned on and stays active" into "The instrument cluster is turned on" and "stays active". The splitting mark matching punctuation would decompose constructions like "X, Y, and Z do something" into "X", "Y", and "Z do something".

Annotation of sentence segments: The annotation of sentence segments as either active or passive or sentence segment without verb is necessary for two reasons:

– For sentence segments without verbs the verbs have to be accordingly adopted from other segments.
– The mapping of grammatical subjects/objects to message senders/receivers is different for active and passive segments.

For the annotation of sentence segments it is possible to use regular expressions based on POS-tags, again. A tagged sentence segment is annotated as passive if and only if it matches the regular expression `".*` $\langle be-form\rangle$`.*VBN.*"` (any character sequence, followed by some form of the verb "to be", followed by a verb participle[4], followed by any character sequence). In this expression $\langle be-form\rangle$ can be equal to "be", "am", "are", "is", "was", "were", or "been". For example, the segment "the_DT instrument_NN cluster_NN is_VBZ turned_VBN on_RP" is annotated as passive because the verb "is" is followed by the participle "turned_VBN".

If the tagged segment does not match the regular expression `".*_VB.*"` (i.e., it does not contain any verb tag), it is annotated as "segment without verb". Otherwise, if the segment contains a verb but does not match any of the passive expressions, the segment is annotated as active, as for example the segment "the_DT driver_NN leaves_VBZ the_DT car_NN".

[3] Here the Java syntax for regular expressions is used. For details see http://java.sun.com/j2se/1.5.0/docs/api/java/util/regex/Pattern.html

[4] verb participle is denoted by the VBN-tag

Extraction of subjects and objects: To extract subjects and objects, active sentence segments are spilt on the basis of the POS-tags into three parts: the verb, the word sequence before the verb, and the word sequence after the verb. Passive sentence segments are split into four parts: the auxiliary verb, the word sequence before the auxiliary verb, the participle, the word sequence after the participle. Then, a previously constructed glossary is used to identify subjects and objects, as in [4]: The grammatical subject is the longest word sequence before the (auxiliary) verb, contained in the glossary. In the case of active segments, the object is the longest word sequence after the verb, contained in the glossary. In the case of passive segments, the object is the longest word sequence after the participle, contained in the glossary.

Propagation of subjects, objects, and verbs: Propagation of subjects, objects and verbs is necessary due to the fact that some sentence segments do not explicitly contain them but share with other segments. The propagation algorithm can be most simply illustrated on the tagged sentence

> "The_DT driver_NN accelerates_VBZ and_CC drives_VBZ faster_JJR than_IN 30km/h_CD and_CC less_JJR than_IN 50km/h_CD ._.",

taken from the instrument cluster specification [3]. This sentence contains three elementary segments:

1. The_DT driver_NN accelerates_VBZ
2. drives_VBZ faster_JJR than_IN 30km/h_CD
3. less_JJR than_IN 50km/h_CD

The first segment contains a subject ("driver") and a verb ("accelerates"). The second segment does not contain a subject but contains a verb ("drives"). Thus, the second segment inherits the subject ("driver") from the first one and results in the segment "driver drives faster than 30km/h". The third segment, in turn, lacks both subject and verb. Thus, it inherits them from the modified second segment and turns into "driver drives less than 50km/h. In a similar way the objects can be propagated as well.

The segments without verb inherit the active/passive annotation together with the verb. When the verb propagation is completed, there are no segments annotated as "segment without verb" any more. This propagation algorithm can be easily generalized to the case where the first sentence segment lacks a verb and also to passive segments. (Generalization not presented here due to space limitations.)

Mapping of subjects and objects to message senders and receivers: When the grammatical subjects and objects have been extracted and the verbs have been propagated to the segments without verbs, it is easy to translate every segment to an MSC message:

active: Message sender equals to the grammatical subject, receiver equals to the object, message content is the word sequence between the verb and the receiver.

passive: Message sender equals to the grammatical object, receiver equals to the subject, message content is the word sequence between the verb participle and the sender.

If the message sender or receiver cannot be identified directly from the sentence segment, they identification is postponed. In this case they are identified by the means of the analysis of the message stack, as in [4].

5 Evaluation: Case Study

The approach presented in this paper was evaluated on the instrument cluster specification [3]. Although not used in an industrial development project, this specification was derived from real industrial documents. This specification was also intended to serve as the contract basis between the car manufacturer and the supplier of the instrument cluster. The glossary, necessary for the translation of scenarios to MSCs, was extracted in our previous work [7].

The case study presented in this paper considered the same set of use cases as the case study performed in our previous work [4]. The difference lies in the treatment of passive and compound sentences. In the case study in [4] passive and compound sentences were manually split and rewritten. Table 1 shows some examples of the performed changes: Sentences in bold font are the corrected versions of the corresponding sentences on the left hand side.

Use Case: activation of the instrument cluster	*Use Case: activation of the instrument cluster*
The driver switches on the car (ignition key in position ignition on).	The driver switches on the car (ignition key in position ignition on).
The instrument cluster is turned on and stays active.	**The instrument cluster turns on.** **The instrument cluster stays active.**
After the trip the driver switches off the ignition.	After the trip the driver switches off the ignition.
The instrument cluster stays active for 30 seconds and then turns itself off.	**The instrument cluster stays active for 30 seconds.** **The instrument cluster turns itself off.**
The driver leaves the car.	The driver leaves the car.

Table 1. Original scenario (left) and corrected scenario used in the previous work [4] (right)

	Matching regular expression	Number of matching sentences
conjunction, and/or	`" [^ ]*_CC "`	96
conjunction, comma	`" [^ ]*_, "`	34
conjunction, colon	`" [^ ]*_:[ ]*"`	23
passive with verb "are"	`".* are_.*VBN.*"`	17
passive with verb "is"	`".* is_.*VBN.*"`	52
other passive forms	modifications of the above expressions	0

Table 2. Case Study: Statistics of Usage of Conjunctions and Passive

The case study consisted of 42 scenarios, containing on the total 384 sentences. Out of these 42 scenarios, only 37 were translated to MSCs in [4]. For the remaining 5 scenarios the necessary rewriting was too extensive. In the original (not rewritten) scenarios, a significant number of sentences contained either passive or conjunctions (cf. Table 2). These sentences were manually rewritten in [4], but treated without any changes

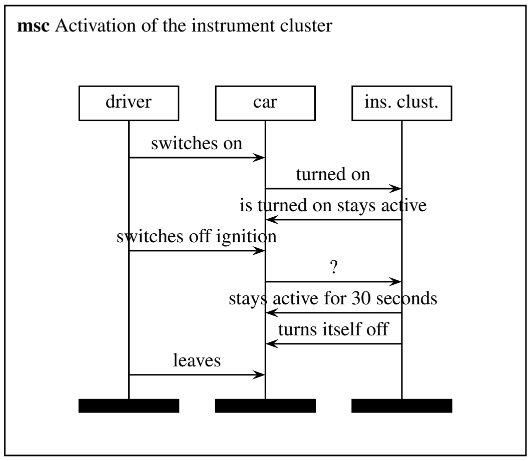

msc Activation of the instrument cluster
driver
car
ins. clust.
switches on
turned on
is turned on stays active
switches off ignition
?
stays active for 30 seconds
turns itself off
leaves

Fig. 2. MSC for the scenario "Activation of the instrument cluster", extracted from the original version of the scenario (left hand part of Table 1)

in the case study presented here. The matchings listed in Table 2 are not disjoint, i.e., there are sentences matching several regular expressions and correspondingly counted in several lines of Table 2. For example, the sentence "The instrument cluster is turned on and stays active" matches both the passive regular expression with the verb "is" and the regular expression for conjunctions with and/or.

To evaluate the correctness of MSCs extracted with the approach presented in this paper, these MSCs were manually compared with the MSCs extracted in [4]. Examples of the extraction results are shown in Figures 2 and 3: Figure 2 shows the MSC extracted from the original scenario (left hand part of Table 1), whereas Figure 3 shows the MSC extracted in [4] from the corrected scenario (right hand part of Table 1). These MSCs are obviously different. This difference results from two facts:

– In Figure 3 two missing messages (marked with "?") are identified, in Figure 2 the first "?"-message is not necessary any more because there is the explicit "turned on" message from the car to the instrument cluster[5]. This difference in MSCs is desirable, as the MSC in Figure 2 better identifies the message flow and makes less guessing of missing messages necessary.
– The MSC in Figure 2 contains a message "is turned on stays active" instead of "stays active". This message name is caused by a tagger error: "stays" is tagged as a noun, thus, the sentence segment "stays active" is considered as a sentence sequence without a verb and inherits its verb from the first sentence segment. Errors of this type can be corrected manually only by the requirements analyst.

Manual comparison of the MSCs extracted with automatic treatment of passive and conjunctions and the MSC extracted in [4] showed that they coincide in 26 cases out

[5] It is easy to use a stemmer, for example the Porter stemmer [8] to convert "turned on" to "turn on". In the presented work this issue was neglected because the grammatical form of the messages is irrelevant for the management of the message stack.

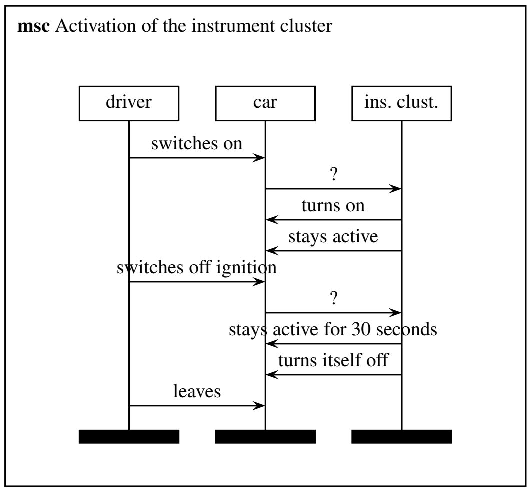


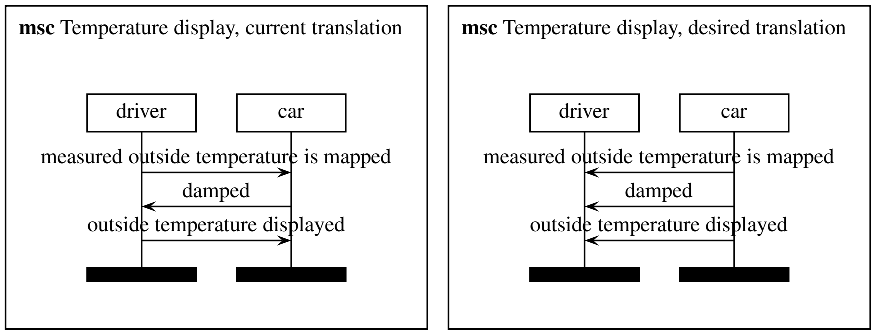
Fig. 3. MSC for the scenario “Activation of the instrument cluster”, extracted from the corrected version of the scenario (right hand part of Table 1) in [4]

of 37 constructed in [4], modulo differences like between Figures 2 and 3. In 11 cases they were different due to the following effect: if all the segments of some sentence are passive, i.e., no message sender can be identified, the algorithm managing the message stack translates these segments to an interleaving sequence of requests and replies, instead of a sequence of messages going in the same direction. For example, the sentence “The measured outside temperature is mapped, damped and outside temperature displayed” is translated to the message sequence in Figure 4(a), whereas the desired translation is shown in Figure 4(b). Manual analysis of the program outputs showed that passive and conjunctions were treated correctly even for such sentences, but the stack management algorithm cannot take sequences of passive sentence segments into account yet. This was not necessary in [4] because the approach in [4] considers active sentences only.

6 Related Work

The idea to use computational linguistic to analyze requirements documents is surely not new. There was a lot of work in this area in recent years. There are three areas where natural language processing is applied to requirements engineering: assessment of document quality, identification and classification of application specific concepts, and analysis of system behavior.

Approaches to the analysis of document quality were introduced, for example, by Rupp [2], Fabbrini et al. [9], and Kamsties et al. [10]. All these approaches have in common that they define guidelines for document writing and measure document quality by analyzing the degree to which the document satisfies the guidelines. These approaches are barely comparable to the approach presented in this paper, as they do not perform any behavior analysis.


msc Temperature display, current translation
driver
car
measured outside temperature is mapped
damped
outside temperature displayed
msc Temperature display, desired translation
driver
car
measured outside temperature is mapped
damped
outside temperature displayed

(a) (b)

Fig. 4. MSCs for the sentence "The measured outside temperature is mapped, damped and outside temperature displayed", translation with the current stack management algorithm (left) and desired translation (right)

Other class of approaches, like for example those by Goldin and Berry [11] and Abbott [12], analyze the requirements documents, extract application specific concepts, and provide an initial model of the application domain. They do not perform any behavior analysis, either.

The approaches analyzing system behavior, as for example those by Ambriola and Gervasi [13], Rolland and Ben Achour [14], Díaz et al. [15], and Vadera and Meziane [16], translate the text to executable models by analyzing linguistic patterns. The approach presented in this paper differs from the approaches by Ambriola and Gervasi and by Vadera and Meziane in one extremely important feature: these approaches analyze only the information directly available in the document and do not reconstruct missing objects and actions. The approach by Rolland and Ben Achour defines rules for manual translation of sentences to messages, but does not perform any automatic analysis. Díaz et al. introduce a transformation technique producing UML sequence diagrams. However, the input to this transformation technique is semantical representation of the sentences and not plain text as in the presented paper.

To summarize, to the best of our knowledge, there is no approach to requirements documents analysis, able to identify missing pieces of behavior, especially by analyzing passive sentences and integrating them in a message stack, yet.

7 Conclusion

Requirements Engineering is a non-trivial task and the presented approach does not claim to solve all its problems. However, it solves several important problems of the early requirements analysis phase:

- It detects missing information in scenarios by guessing message senders/receivers in passive sentences.

– Compared to [4], it makes rewriting of passive and compound sentences unnecessary.
– It translates textual scenarios to MSCs, allowing for further validation.

When validated, the constructed MSCs can be used in further software development. Thus, the approach presented in this paper makes a contribution to behavior modeling. As shown in a case study, the approach is applicable to industrial documents.

References

1. Mich, L., Franch, M., Novi Inverardi, P.: Market research on requirements analysis using linguistic tools. Requirements Engineering **9** (2004) 40–56
2. Rupp, C.: Requirements-Engineering und -Management. Professionelle, iterative Anforderungsanalyse für die Praxis. Second edn. Hanser–Verlag (2002) ISBN 3-446-21960-9.
3. Buhr, K., Heumesser, N., Houdek, F., Omasreiter, H., Rothermehl, F., Tavakoli, R., Zink, T.: DaimlerChrysler demonstrator: System specification instrument cluster (2004) http://www.empress-itea.org/deliverables/D5.1_Appendix_B_v1.0_Public_Version.pdf, accessed 11.01.2007.
4. Kof, L.: Scenarios: Identifying missing objects and actions by means of computational linguistics. (2007) Contribution to the 15th IEEE International Requirements Engineering Conference.
5. Grosz, B.J., Joshi, A.K., Weinstein, S.: Centering: A framework for modeling the local coherence of discourse. Computational Linguistics **21** (1995) 203–225
6. Ratnaparkhi, A.: A maximum entropy model for part-of-speech tagging. In Brill, E., Church, K., eds.: Proceedings of the Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, Somerset, New Jersey (1996) 133–142
7. Kof, L.: Text Analysis for Requirements Engineering. PhD thesis, Technische Universitaet Muenchen (2005)
8. Porter, M.: An algorithm for suffix stripping. Program **14** (1980) 130–137 http://www.tartarus.org/ martin/PorterStemmer/, accessed 14.07.2003.
9. Fabbrini, F., Fusani, M., Gnesi, S., Lami, G.: The linguistic approach to the natural language requirements quality: benefit of the use of an automatic tool. In: 26th Annual NASA Goddard Software Engineering Workshop, Greenbelt, Maryland, IEEE Computer Society (2001) 97–105
10. Kamsties, E., Berry, D.M., Paech, B.: Detecting ambiguities in requirements documents using inspections. In: Workshop on Inspections in Software Engineering, Paris, France (2001) 68 –80
11. Goldin, L., Berry, D.M.: AbstFinder, a prototype natural language text abstraction finder for use in requirements elicitation. Automated Software Eng. **4** (1997) 375–412
12. Abbott, R.J.: Program design by informal English descriptions. Communications of the ACM **26** (1983) 882–894
13. Ambriola, V., Gervasi, V.: The Circe approach to the systematic analysis of NL requirements. Technical Report TR-03-05, University of Pisa, Dipartimento di Informatica (2003)
14. Rolland, C., Ben Achour, C.: Guiding the construction of textual use case specifications. Data & Knowledge Engineering Journal **25** (1998) 125–160
15. Díaz, I., Pastor, O., Matteo, A.: Modeling interactions using role-driven patterns. In: RE'05: Proceedings of the 13th IEEE International Conference on Requirements Engineering (RE'05), Washington, DC, USA, IEEE Computer Society (2005) 209–220
16. Vadera, S., Meziane, F.: From English to formal specifications. The Computer Journal **37** (1994) 753–763