Enhancing Coherency of Specification Documents from Automotive Industry

Jean-Noël Martin[1] and Damien Martin-Guillerez[2]

1 **All4Tec**
6 Rue Léonard de Vinci – BP 0119 – F-53001 LAVAL Cedex, France
jnm@all4tec.net

2 **Inria Bordeaux Sud-Ouest**
351 cours de la Libération, 33405 Talence Cedex
damien.martin-guillerez@inria.fr

Abstract

A specification describes how a system should behave. If a specification is incorrect or wrongly implemented, then the resulting system will contain errors that can lead to catastrophic states especially in sensitive systems like the one embedded in cars.

This paper presents a method to construct a formal model from a specification written in natural language. This implies that the specification is *sufficiently* accurate to be incorporated in a model so as to find the inconsistencies in this specification. *Sufficiently* means that the error rate is down 2%. The error counting method is discussed in the paper. A definition of specification consistency is thus given in this paper.

The method used to construct the model is automatic and points out to the user the inconsistencies of the specification. Moreover once the model is constructed, the general test plan reflecting the specification is produced. This test plan will ensure that the system that implements the specification meets the requirements.

1998 ACM Subject Classification I.2.7 Natural Language Processing

Keywords and phrases coherency, specification, model generation, automatic text processing

Digital Object Identifier 10.4230/OASIcs.SLATE.2012.225

1 Introduction

Specification documents define the way a system should behave and not how it is structured. They list the system's properties and limits that we will call qualifiers. Those documents are used to communicate between stakeholders of a project and are the reference documents to verify if an engineered system meets the requirements. To ensure the correctness of the system, modeling its specification with a formal paradigm such as Markov chains is of great importance. It indeed enables automatic tests [10].

Of course, specification documents contain errors, leading to defects in the design and thus in the product itself. However, determining whether a specification is correct or not is a hard job especially due to lack of criteria to judge whether a specification is correct or not. The IEEE has issued standards on the subject such as the IEEE 830 [1] and the IEEE1233 [2], but these standards provide only part of the criteria and no objective criteria to evaluate a specification. In fact, they define general rules but do not give support to a quantitative qualification nor for a complete list of qualities.

To determine objective criteria to judge the correctness of a specification, one must define what the means for a specification are to be correct. Of course, consistency is a key aspect of

1st Symposium on Languages, Applications and Technologies (SLATE'12).
Editors: Alberto Simões, Ricardo Queirós, Daniela da Cruz; pp. 225–237

OpenAccess Series in Informatics
OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

correctness. Correctness includes both a good choice of words and the consistency itself. If a specification is consistent, then the construction of a formal model of the specification becomes possible In this paper, consistency is taken into account in our work and we automatically extract a Markov model for test generation software from a specification document from the automotive industry that is written in English. However a wrong choice of words will be reported in the model will show that there will not be any negative consequences. This paper is organized as follows: Section 2 presents the work related to this study and Section 3 presents the key aspects of specification documents that enable text analysis. After an overview of the method used in Section 4, we go through the several steps of the treatment of the specification in Section 5. We then examine how to improve the consistency of a specification document thanks to our system in Section 6. Finally, we conclude in Section 7.

2 Related Work

Since the work done by Chomsky in 1965 [7] on formal grammars, automatic language processing has evolved. Of course grammar parsing has also evolved. We can notably mention the works of Nique [18] and Winograd [23] on context-free grammar. More recently a new book had summarized the state of the art that was produced by Jurafsky and Martin [11]. Nique and Winograd explain in a clear way what the Chomsky approach is. Jurafsky opens on the use of Markov models. These analyses, from Winograd to Juravsky, provide a large base of information on how to process documents analysis in natural language and extract relevant subparts of a document. Other works of interest are based on the concept of translation as explained by Lutkens and Fermont [15], Planas [19]; especially those to compare text [20] using alignment techniques. However, no work has really succeeded in generating general purpose parser of natural language. Nevertheless, specification documents are well structured and we will show in this paper that it is possible to parse them, to provide accurate detection on non consistency, using some adaptation of the Chomsky grammar mixed with alignment techniques.

Extracting the model from this parsing requires having a consistent specification at the entry of the system. However, it is not clear what a consistent document is.

The consistency domain has been explored by Michel Charolles [4, 5] and by Fabien Wolf [24]. The sentential calculus is addressed initially by Michel Charolles [6] as the consistency theory. The world of uncertainty is discussed by R. Martin [17] and Lehrer [14] and deals with fuzzy logic that give a track to the alignment of texts. The concept of 'discourse framing', i.e., generating a tree structure giving a summary of the speech, comes from the work of Robert Martin [17] which relied on G. Fauconnier's work [8]. All those works give key points of what makes a text consistent but we found no work that precisely defines the component of consistency of a text. So we did it.

Actually, some research has addressed the topic of making a specification consistent with various results. Out of those, we can quote V. Gal [9], P. Serre [21] and M. W. Trojet [22]. All those works rely on the description of the specification in specific language like the language Z [15, 16] that is far from natural language.

This paper leverages the existing works on grammar and translation systems to parse the well-structured specification documents.

As an example of what we call a well structured document, Listing 1 exhibits a paragraph of a final writing of a specification. This start with a title which defines the universe and any of the above requirements include one or two preconditions linked to a predicate.

This paper also presents a definition of the six qualities that compose consistency and

Listing 1 An extract of a specification.

```
Speed
[DRL004-A-v1] the boot lid release feature shall be enabled by any
 boot lid release button, if the ignition is run or start and the
 vehicle speed is no more than 7 km/h.
[DRL004-A-v1] the boot lid release feature shall be enabled by any
 boot lid release button, if the ignition is off, independent of the
 actual vehicle speed information.
[DRL004-A-v1] the boot lid release feature shall be enabled by any
 boot lid release button, if the vehicle speed fault bit is set,
 independent of the actual vehicle speed information.
```

how the parsing uncovers consistency errors in specification documents. Indeed, Indeed, the previous contributions consisted mainly in giving example of what was an inconsistent text. We never found any definition of consistency except the one included in the dictionary.

Finally, this parsing results in a Markov model that can be used to prepare the test plan for the system.

3 Specification Structure

We based our work on four specification documents written in English from the automotive industry. Two of them come from Lear, the next comes from Audi and the last comes from BMW. From those documents, we extracted the overall form of a specification document. It is composed of four different semantic elements:

- an action (or predicate), denoted P, describes what is done (for example, *"the user presses the button to open the device"*, *"the user presses the button xxx that starts the motor that opens the boot lid"*), this defines the post-condition;
- a condition, denoted C, defines the initial conditions under which the predicate may apply (for example, *"when the vehicle is traveling slower than 7 km / h"*, *"when the Can message is not missing"*);
- a property, denoted Q, is a group of terms that gives quantitative data on the action (for example, *"during T_EnableBootLidRelease = 23 seconds"*, *"PJB_RearWindowsStatus = 0 within the Can message 433"*);
- and some background information, denoted A, treated as comments.

Thus, we consider that a specification is a sequence $(C+\ P\ Q*\ A*)+$[1]; in a specification one requirement is a set of non empty preconditions that applies to an action; this action is qualified by one, several or no property and is subject to potential comments. A specification is a list of requirements.

4 System Overview

We built software that convert a consistent specification written in English into a Markov model for test generation software. This system is based on the fact that specification documents are well-structured as presented in the previous section. This software is decomposed in 7 steps as shown in Figure 1:

[1] For clarity the + sign means one or more and the ∗ sign means zero, one or more.

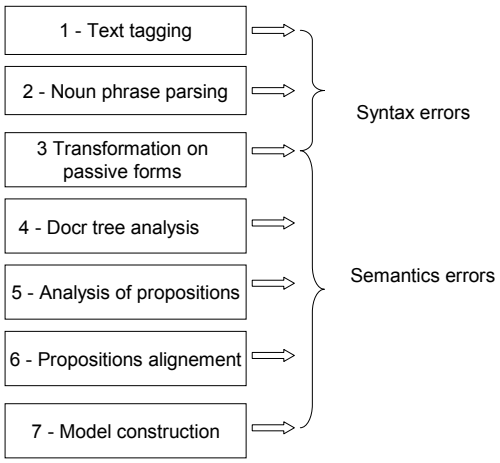

Figure 1 System overview.

1. A text tagging phase introduces the process, including the choice of stating the value of words in conjunction with a statistical tagger [3].
2. A text parsing step, based on the Chomsky grammar [7, 12, 13, 18], decomposes the text into word phrases, that is into elementary groups of words that have a grammatical function.
3. The sentences that are in passive forms are then transformed into active form.
4. The doc tree elements are added to the propositions which are categorized in $PQCA$ propositions.
5. The sentences are decomposed into three term propositions (subject / verb / complement) (see the work of F. Wolf [24]).
6. The propositions are aligned using classical translation methods. This enables the identification of similar terms (i.e., the same action or the same object) written differently in the text.
7. Finally, the propositions verbs are converted into states of a Markov Chain and the logic, extracted thanks to steps 2 and 5, is converted into transition between states. Subjects are the inputs and complements are the expected result.

This process succeeds if and only if most consistency errors have been removed from the specifications. From the experiment explained further we had an automatic diagnostic of 97.6% of errors found. This condition enables automatic detections of many consistency errors especially ambiguous statements in the specification.

5 System Description

This section presents each step done to parse a textual specification into a Markov Model.

5.1 Text Tagging

The text tagging tags the words of the documents with a set of 23 parts of speech. This is done thanks to Brill parser [3]. At this stage we initialize the value of words: the value of words is an extension of the excluded attribute given to weak words (weak words are the

Listing 2 Definition of the grammar of a proposition.

```
<C> => <triggers word><SN>
        | <triggers+> <Text><SV><(<pivot><Text>)
<P> => <SN> + <SV>
        | <GV> <SN><SV>
<Q> => <CONJ><SN>
        | <SN>
<Text>  => <SN><SV >
        | <SN>
        | <SV>
<SN> => <DET><N+><O>
        | <PREP> <SN>
        | <Past participle>
        | <DET><A><N+>
        | O
        | <DET><NP>
<SV> => <Aux><GV>
        | <GV>
<GV> => <V><Q>
```

words that have a poor semantic value: the articles, the auxiliary, etc...). The value of words consists of allocating a value to any words, mainly:

- 0 for weak words;
- 1 for normal words;
- 3 for numeric values;
- 5 for named entities;
- 8 for negative words.

The allocation of the value of word at the tagging stage prepare for a further accurate alignment of propositions. The value of word is chosen to provide a first idea of how to quote the word depending on the semantic discrimination of any word.

5.2 Text Parsing

As we explained before, a specification is composed of conditions C, predicates P, properties Q and comments A. As a consequence, our system leverages the regular form of specification to analyze it. The parsing is thus the extraction, for each specification line, of the C, P, Q and A elements.

This extraction is a classical grammar parsing using the grammar detailed in Listing 2. To illustrate this schema we express the first rule in words: a condition C is made of a trigger word that introduces a nominal noun phrase or a trigger that introduces a text followed by a verb phrase, etc.

This grammar is a sub part of the one proposed by Chomsky in [7] to include only part relative to the English used in the specification document. It specifies how a condition C, a predicate P, and a property Q are decomposed. Figure 2 shows the results of the parse tree of the sentence *"If the PJB is configured for S389 and receives a valid boot lid release request from the CRM"*. Of course the root of the parse tree in that figure is a condition C with the *"if"* word as the trigger. *"the PJB"* is the subject to the *"is configured for S389"* verbal group. *"and"* is a pivot introducing the text *"receives a valid boot lid release request from the CRM"*. Using this grammar, we thus have a basic decomposition of the text.

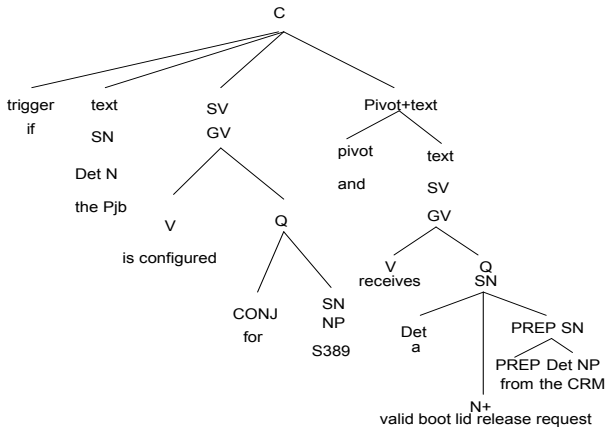

Figure 2 Grammar parse tree of a condition.

Listing 3 TPassive rule from the transformation defined by Chomsky.

```
TPassive:
SN1 + Aux + "to" + VPP + SN2 <=> SN2 + Aux + "be" + VPP + SN1
```

5.3 Active Form

Once the sentences are extracted using the previously described grammar, sentences in passive form are transformed into active form. This step is performed so that logic of the specification is direct and thus transferable to a Markov model.

First of all we stated that the transformations were dual. Assuming VPP stands for verb participle, which is changed to infinitive thanks to the rule exhibited in Listing 3. This rule comes from the Chomsky defined transformation but it is limited to the only sentence with exactly two verbs. We extended it with two new rules shown in Listing 4[2] to include all the possible cases especially sentences with three verb phrases and simpler sentences with one only verb phrase.

5.4 Doc Tree Adding

Thanks to the plan of the document we extract the "Universe" from the document and we construct a couple of propositions, adding a "Universe" to any proposition. The "Universe" concept comes from the works of R. Martin[17] and G. Fauconnier [8]. In our case we extract the universe from the plan of the documents.

The "Universe" step is the one where the major traps for incorrect statement are set.

[2] In Listing 4, noun phrases are labelled $GN1$ to $GN4$ and verb phrases VPP, $VP1$ and $VP2$

Listing 4 Grammar for passive forms with three verbs.

```
<VPP> <GN1> <VP1> <GN2> <GN3> <VP2> <GN4>
   <=>
<GN3> <GN4> <VP2> <"to"> <VPP> <GN1> <"to"> <VP1> <GN2>
```

5.5 Proposition Normalization

As shown in Figure 1, sentences are then decomposed in propositions by analyzing the parse tree obtained by the second step. Each proposition is a group of three groups: a subject, a verb and a complement. Each proposition is thus stored in that form and the logic between propositions is kept (for example, the *"if"* trigger of the sentence analyzed in Figure 2 is kept as a causal link between the sentence analyzed and the following sentence). The two propositions are *"the PJB is configured for S389"* and *"the PJB receives a valid boot lid release request from the CRM"*. The proposition analyzed in Figure 2 shows after that step:

1. *"the PJB"* (subject) *"is configured"* (verb) *"for S389"* (complement)
2. *"the PJB"* (subject is to be found) *"receives"* (verb) *"a valid boot lid release request from the CRM"* (complement)

The two propositions are expressed in the causal links *"if 1 and 2"*. Note that the subject of the second proposition needs to be extracted from the first proposition. After this step, we have a list of propositions decomposed into subject / verb / complement and the causal link between each proposition.

5.6 Proposition Alignment

The subsequent step is to align the propositions using translation techniques. This step helps to identify propositions or nominal groups that have the same meaning but that are written differently.

To do so, we use alignment techniques described by Lutkens et al. [15] and Planas [19]. To measure the similarity between sentences, the Jaccard measure is used [20] because it is simple and meets our needs. Used in conjunction with the value of words, it allows for an accurate comparison between propositions: for example two propositions with a different numerical value will be paired and if a negative word is in one of the propositions the alignment will fail. Thanks to the same propriety we can correct a wrong comma in a sentence. Calculating the Jaccard value with and without the value of word will disclose formal contradiction.

5.7 Markov Model Generation

Using the previous steps, we now have a list of propositions with their causal link in a database.

Using this, we can construct the Markov model for test generation software. The model for this software is described in [10] and was chosen for our process because it has proven its efficiency for test plan generation.

A Markov process is defined by a process in which the probability of going to a given state only depends on the current state. The direct consequence of this property is that a Markov process can be described as a series of states and a transition function. This function $\delta(x, y)$ gives the probability of transiting from state x to state y. The model used by the test generation software is a variation of Markov processes in which the transition functions are also labelled by an extra condition that should be met for the transition to be triggered (Figure 3).

This model thus represents perfectly the elements extracted from the specification. Each proposition is a state and the causal link a transition. The extracted model (Figure 4) can then be inserted into the test generation software that will construct the test plan for the specified system.

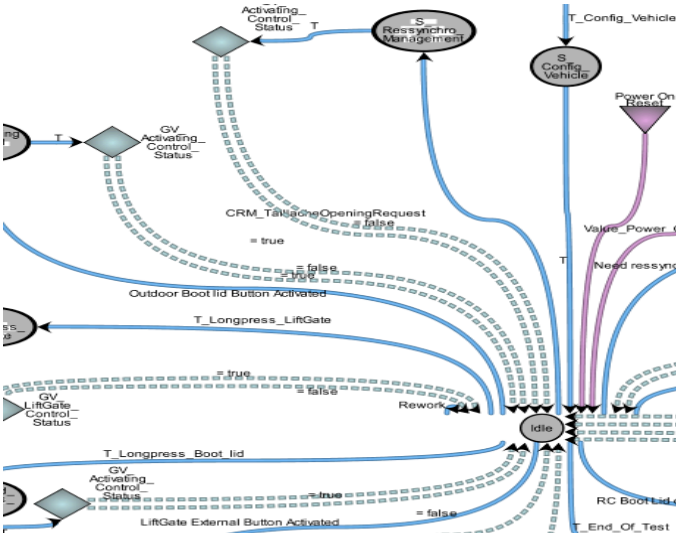

Figure 3 Extract of a manually constructed model.

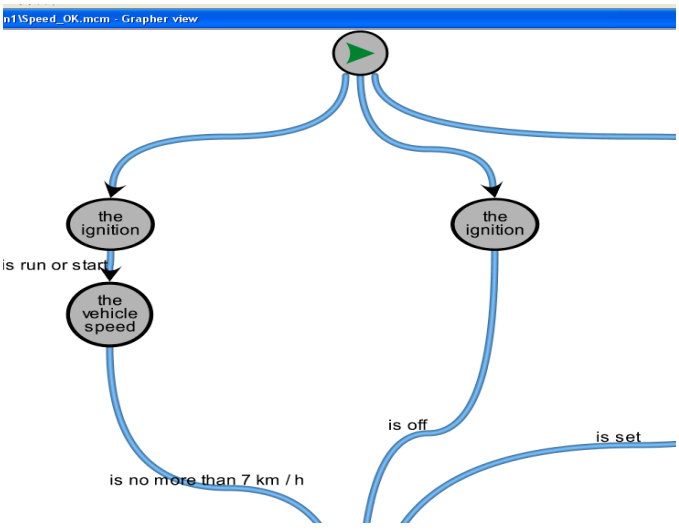

Figure 4 Extract of a manually constructed model.

6 Enhancing Consistency helped by the Model Generator

As presented in Figure 1, several steps in the model generation enable the detection of errors in the specification documents. In this section, we present how our system can be used to remove several consistency errors in specifications after presenting our definition of the consistency of a specification.

6.1 Consistency Approach

The word consistency distinguishes parts of an ensemble that focuses on a logical connection including a lack of contradiction in the devoted set; these parts are closely linked. They include a logical link and are organized to provide a stepwise link. This consistency initially dedicated to the discourse, proved suitable for the specification documents.

The first quality that we require from a text, related to consistency, is the property of a text to make sense. It is easy to infer informational relation between elements from one part of a document to the next. We will make use of the example provided by a personal access system equipment to demonstrate this property:

- The *"Pase"* is the smartcard that allows a vehicle user to open the vehicle doors and to click on the switch instead of the ignition key;
- The *"Pase"* user can approach his vehicle without taking the smartcard in his hands;
- He can see the flashers welcome him, and he can hear the door unlock;
- After entering the vehicle, he uses the *"Pase"* which he introduces into a smartcard reader, and he can then press the *"Start"* button to start the vehicle.

The inferences here are common, respecting the principle of relevance in the transmission of information. The inferences to be done to understand this text are progressive. The *"Pase"* replaces the ignition key, and there is a button that replaces the *"Neimann"*. Yet it must characterize these inferences, "the *"Pase"* is the smartcard (...) instead of the ignition key" and thus explicitly states the first inference, and "the *"Start"* button (...) starts the vehicle" explicitly states the second inference. Further progress is respected: we discussed the device, then we talk about the doors opening and finally about the car starting.

Remove the first sentence from this text and we can no longer justify its consistency. We must then make a less explicit inference. "The *"Pase"* is a remote command."

Note that the principle of relevance is more demanding in a written language when the preset is low than in the spoken language where the preset is important because of the knowledge of the listener. It must be recognized that many texts do not satisfy this property. We have many examples that give situations of ambiguity in the texts:

- In the text *"The phone rings. I drive my car."*, there are some connections to be established to achieve consistency.
- The text *"The vehicle was parked by Mark on a very busy place. The noise was terrible. Paul spent the evening on a bench beside the ocean. The wind was blowing. It was raining."* is given as an example of ambiguity: submitted to a panel of readers, interpretations range from *"there is a vehicle parked in a noisy place and strangely a man named Paul spends an evening at the seaside"* to *"Paul, who was in the camper is sad to stay there the next day because of bad weather"*.

6.2 Definition of the Consistency

We identified a series of documents qualities that make the documents understandable. Some of them are components of the documents consistency while others simply make the

documents less ambiguous and more understandable.

The qualities of documents that do not qualify as components of the consistency are:

- The undefined or ambiguous words: words that are not defined in the dictionary are to be defined or changed, a word in a text that combines a single definition in a given context is not ambiguous.
- The generic documents deserve to be written in the generic present tense, standardizing as much as possible the negative forms[3].

The qualities of the documents that qualify as components of the consistency:

- Cohesion and progressiveness: cohesion and progressiveness are the properties of a text that establish the continuity of the progress of the text, the property reflects the ability of text to be consistent in terms of chronological steps;
- Logical consistency: we define logical consistency as the absence of logical contradictions raised by the text;
- Clarity: we will define clarity as the property of a group of proposals to mean something in a clear way;
- Plausibility: the plausibility of an act is its ability to seem possible. In the field of natural language, we will consider as a plausible sentence a phrase that we are not surprised to hear [8]. Operationally we are considering the Dempster and Shaffer's theory, which can allocate two trust values to a predicate such as credibility of P defined by $Credibility(P) = 1 - Confidence(\neg P)$;
- Explicit knowledge: knowledge is explicit if it helps to understand a text without knowing the local context; it is based on the principle of relevance applied to items overlooked by the author in the specification;
- Accuracy of the text consists in lack of over information: we found in some texts two parts of the text which have exactly the same meaning. In the specification domains, this is named over specification.

Consistency of specification documents will be defined in part by our ability to generate a test model that can produce a test plan. The deficiencies in this criterion will be considered as inconsistencies in the text and we will work to detect and fill them with the cooperation of the author of the specification document.

If consistency requires a precise definition, it is not difficult to define the inconsistency: in fact the inconsistency is the property of a text whose consistency cannot be established.

6.3 Removing Inconsistencies in Specifications

We believe that our automatic translation of specification documents into the Markov model helps to remove most consistency errors in specification documents (we measured 97,6% consistency errors removed in our specification documents compared to manual consistency error removal).

The parsing of the text detects grammar errors and some ambiguous words. The user is prompted to define ambiguous words or to change them. We rewrites the document in the generic present tense. Defects related to cohesions and progressiveness are found at the universe step. At the model step, the absence of links (non-attainable states) could also disclose lack of logical consistency. The proposition alignment uncovers ambiguous and undefined propositions by duplicating links in the model. Finally, the relevance of the

[3] This quality is relevant for specification documents but seems to not apply to novels

Table 1 Table of defects automatically found. The defect rate is stated as the total weight of the defects rated to the number of words of the document.

	First specification			Second specification		
diagnosis	**occurrence**	**weight**	**total**	**occurrence**	**weight**	**total**
lexical conformity	148	1	148	18	1	18
missing words	14	1	14	2	1	2
text parts	4	3	12	0	3	0
syntax not conform	6	9	54	20	9	180
progressiveness	1	96	96	0	96	0
logical inconsistency	2	18	36	1	18	18
lack of clarity	0	36	0	1	36	36
explicit knowledge	3	9	27	7	9	63
accuracy	1	9	9	2	9	18
defect total weight			396			335
size of the document			1172			4440
defect rate			33,79%			7,55%

specification is hard to assess but the obtained model will be intricate if the text is irrelevant. Indeed, irrelevant phrases induce extra states and links in the model.

We applied those principles to two specifications from the automotive industry and it leads us to a specification that we believe is non ambiguous and permits the extraction of the model shown in Figure 4. In the automatic analysis we have an automatic diagnosis of non-correct writing and from that we issued the Table 1. This table shows the number of occurrences of a class of error, the estimated weight of a error (i.e., the number of modification the error requires to fix it) and the total weight of errors.

The defects extracted are the result of the whole process. The main parts are detected at the analysis step, mainly at the "Universe" step, but some errors are detected at the model step due to the model form that make the errors evident.

Finally we can say that the three major classes of error are balanced: a third is semantic (41%), a third is syntaxic (33%), and the remaining part is lexical. Among the semantic errors the main part is progressiveness then explicit knowledge, then logical consistency then clarity, and finally accuracy.

7 Conclusion

In this paper, we presented our method for automatically extracting a Markov model for a test generation software from a specification written in english using a natural language. We proved this method works by applying it to a real specification from the automotive industry. This method is now to be applied to large specifications up to 500 pages.

The principle that has been applied to automotive documents should soon be applied to other disciplines like defense or transportation.

This work is made possible thanks to the large literature in automatic language processing and to the structure of specifications. It is now being integrated into the MaTeLo software and we are extending it to handle other languages: French is soon to be used, Italian, Spanish and Portuguese are natural but German introduces specific problems.

This model extractor also help to make a specification consistent. We defined what we call the consistency of a document that is to say a document has to adhere to six properties. Each

step in the automatic analysis can disclose consistency problems and errors in the obtained model. This discloses the most dramatic consistency problems of the original specifications. Once the specification is consistent enough, the obtained model will generate the test plan for the final system. This system will increase the quality (less defects) of the resulting products.

A consistent specification might still need repetition of explicit knowledge to ensure that the reader who will implement the specification does not overlook important information. Ensuring such a thing in a current work will increase the overall consistency of specification documents. A last improvement in our roadmap is to address non-functional properties specified in the document. For instance, a specification can contain safety rules to avoid a certain state and those rules should be translated into the model. Our current implementation needs to be improved to handle those general rules.

Acknowledgements Jean-Noël Martin would like to thank his Thesis advisors very much: Bernard Levrat and Amghar Tassadit due to the fact they convinced him to do this project as it will be of benefit to his company.

References

1 IEEE 830 – recommended pactice for software requirement specifications, 1993.
2 IEEE 1233 – guide for developing system requirement specifications, 1996.
3 Eric Brill. A Simple Rule-Based Part Of Speech Tagger. In *Third Conference on Applied Computational Linguistic*, 1992.
4 Michel Charolles. Note sur la cohérence des textes. *Pratiques*, 10:105 – 111, 1976.
5 Michel Charolles. Text connexity, text coherence and text interpretation processings. In E. Sozer, editor, *Text Connexity, Text Coherence, Aspects, Methods, Results*, pages 1 – 16, 1985.
6 Michel Charolles. Cohérence, pertinence et intégration conceptuelle. *Intégration Conceptuelle*, 2002.
7 Noam Chomsky. *Aspects of the theory of syntax*. MIT Press, 1965.
8 G. Fauconnier. Domains and connections. *Cognitive Linguistics*, 1:151–174, 1990.
9 Viviane Gal. *Spécification à l'aide du langage LOTOS d'un algorithme de gestion d'une mémoire répartie à cohérence causale*. CNAM, 1995.
10 H. Le Guen and T. Thelin. Practical experiences with statistical usage testing. In *Eleventh Annual International Workshop on Software Technology and Engineering Practice*, 2003.
11 Daniel Jurafsky and James H. Martin. *Speech and Language processing*. Pearson International Edition, second edition, 2009.
12 J. J. Katz. *Sur la compréhension des phrases agrammaticales.*, chapter Semi-sentences, pages 400–416. Fodor and Katz, 1964.
13 J. J. Katz and P. Postal. *An Integrated Theory of Linguistic Descriptions*. MIT Press, 1964.
14 Keith Lehrer. Coherence, consensus and language. *Linguistics and Philosophy*, 7(1):43 – 55, 1984.
15 E. Lutkens and Ph. Fermont. A prototype machine translation based on extracts from data. Manuals, Université libre de Bruxelles, Bruxelles, 1985.
16 Clara Mancini, Donia Scott, and Simon Buckingam Shum. Visualing discourse coherence in non linear documents. *TAL*, 47(2), 2006.
17 Robert Martin. *Pour une logique du sens*. Presses Universitaires de France, 1983.
18 C. Nique. *L'initiation Méthodique à la grammaire générative*. Colin, 1974.
19 Emmanuel Planas. *Structure et algorithmes pour la traduction fondée sur la mémoire*. PhD thesis, Université Joseph Fourier, Grenoble, 1998.

20 Reinhard Rapp. Identifying word translations in non-parallel texts. In *the 33rd annual meeting on Association for Computational Linguistics*, 1995.

21 Philippe Serré. *Consistency of the geometrical specification for objects in n-dimensional Euclidian space*. PhD thesis, École Centrale des Arts et Manufactures, 2000.

22 Mohamed Wassim Trojet. *Approche de vérification formelle des modèles DEVS à base du langage Z*. 2010.

23 T. Winograd. *Language as a Cognitive Process*, volume 1: Syntax. Addison-Wesley, 1983.

24 Fabien Wolf and Edward Gibson. *Coherence in Natural Language*. Massachusetts Institute of Technology, 2006.