Mining incoherent requirements in technical specifications: Analysis and implementation

Patrick Saint-Dizier

CNRS, IRIT, Toulouse Cedex, France

Keywords:
Requirement engineering
Linguistics of requirements
Incoherence analysis
Natural language processing

A B S T R A C T

Requirements are designed to specify the features of systems. Even for a simple system, several thousands of requirements produced by different authors are needed. Overlap and incoherence problems are frequently observed. In this article, we propose a method to construct a corpus of various types of incoherences and a categorization that leads to the definition of patterns to mine incoherent requirements. We focus in this contribution on incoherences (1) which can be de-tected solely from linguistic factors and (2) which concern pairs of requirements. Together, these represent about 60% of the different types of incoherences; the other types require extensive domain knowledge and reasoning. The second part of this article develops several language-based patterns to detect incoherent requirements in texts. An indicative evaluation of the results concludes this contribution. More generally, this contribution opens new perspectives on in-coherence analysis in texts.

1. Motivations and objectives

Requirements (or business rules) [11] form a specific class of documents with specific functions: they are designed to describe a task, a system, a product, a regulation or any kind of situation in a declarative way [1,3]. Requirements do not explain how to realize a task: they state properties that a process or a product should have or constraints that they should meet. For example, in the case of an information retrieval system:

(1) *Any query must get a response in less than 2* s

states an efficiency constraint, but does not say how to realize it. Furthermore, a requirement must state atomic or unique constraints. In our previous example, additional requirements must be written if there is also a constraint e.g. on the length of the response:

(1a) *Any query must get a response in less than 2* s
(1b) *The response to any query must be shorter than 50 words.*
(1c) *The response to any query must be shorter than 5 sentences.*

Requirements must describe in an as comprehensive as possible way the features a product should have or the way an organization should be managed via rules. As a consequence, even for a small product, several thousands of requirements are often necessary. For example, the wing structure of the Airbus A380 is described by more than 180 000 requirements, hierarchically

E-mail address: stdizier@irit.fr.

https://doi.org/10.1016/j.datak.2018.05.006

organized in different modules.

A requirement may be composed of a single statement or may be associated with one or more supports that justify it. These supports are usually called the 'rationale' of the requirement. In some specifications, warnings describe the consequences of not following requirements, while advice describe optional requirements that may improve a result, e.g. the design of a product. Both warnings and advice are specific forms of arguments [5].

Requirements must follow precise authoring guidelines, which may depend on the domain and also the target reader or user, e.g. Refs. [15,26]. The Lelie system [20,21] aims at detecting authoring errors in requirements. In spite of precise and easy to use authoring guidelines, Lelie shows that unclear, ambiguous, incomplete or poorly written requirements occur relatively frequently. A rate of 20% of incorrectly written requirements is frequently found in requirements which have been validated by experts. This results in a significant waste of time to understand a specification, in difficulties to update, trace and re-use these specifications. More importantly, there are risks of misconceptions leading to incorrect realizations or exposure to health or ecological problems. Controlling how requirements are written is a high and fast return-on-investment activity. For example, in the maintenance sector, poorly written requirements can entail extra costs up to 80% of the initial product development costs.

The different protagonists involved in requirement production is a source of mismatches, inconsistencies and redundancies: stakeholders, technical writers, managers, users and manufacturers may all play different roles and have different views on the requirements. These discrepancies are developed in e.g. Refs. [23–25]. To overcome these problems, most industrial sectors have defined authoring recommendations, methods and tools (e.g. Doors), to elaborate, structure and write requirements of various kinds. The result is an easier traceability, control and update. However, our observations of technical authors at work show that those recommendations, in spite of the existing tools, cannot strictly be observed in particular for very large sets of requirements.

Authoring tools have emerged two decades ago. The first tool was developed and tested at Boeing [26], then at IBM with Acrolink and Doors, and, more recently, Attempto [6] that has some reasoning capabilities. Grady [10] and Asooja et al. [2] among many others, developed a number of useful methodological elements for authoring technical documents. A synthesis of controlled language principles and tools is presented in Refs. [14,15]. Most of these principles and tools have been customized to requirement authoring, based on general purpose or domain-dependent norms (e.g. INCOSE, IREB or ISO26262). The difficulty that remains is recurrent: the customization of these tools to the author profile, the domain authoring practices, and the type of requirement.

Several road-maps on requirement elicitation and writing, e.g. Wyner et al. [27], show the importance of having consistent and complete sets of requirements, and rank it as a priority. However, no concrete solution so far, to the best of our knowledge, has been developed to characterize the coherence problem. Projects such as those developed by Kloetzer [13] at finding contradictions between lexico-syntactic patterns in Japanese, including spatial and causal relations. This is however quite different from the problem that is addressed here, since inconsistencies may have very diverse forms (section 4). One of our main aims is indeed to characterize these forms w.r.t. requirement authoring principles. In De Marneffe [18], the limits of textual entailment as a method to detect inconsistencies is shown and a corpus is developed from which a typology of contradictions is constructed. The need of a very fine granularity in the data is advocated to avoid errors, which is not possible for large sets of requirements.

Finding out incoherences in specifications is recognized by requirement authors to be a crucial and a very hard problem. Of interest is the site: www.semanticsimilarity.org that offers several tools for measuring similarities in texts. Also of interest are the extraction patterns defined in Ref. [8]. The use of SAT solvers, which are satisfiability solvers, running efficiently and in a sound way on very large sets of propositions, can be foreseen to detect deeper forms of inconsistencies where knowledge and logical aspects are involved. However, this means translating requirements into propositional logic or into Horn clauses. Both of these translations fail to capture several facets of requirements, in particular complex VPs, modals and the discourse structure that frequently adds restrictions to a requirement. Our goal is to instead develop specific linguistic patterns that can identify incoherences between requirements. These patterns could be viewed, possibly, as instantiated SAT templates.

In this article, we first show how a corpus of incoherent requirements can be constructed and annotated. This corpus leads us to develop a categorization of inconsistencies based on linguistic considerations. Finally, we show how language patterns can be developed and evaluate the results obtained at this stage. This contribution opens new perspectives (1) on new forms of incoherence mining in texts and (2) on mining incoherent arguments more generally. The construction of this corpus is extremely challenging: large volumes of requirements are necessary, which experts are usually not willing to share. Furthermore, incoherent requirements are in general not adjacent in a text: this makes incoherence mining quite costly in terms of computation since it is necessary to compare each requirement with other requirements which may be in other sections or in different documents. To the best of our knowledge this is the first attempt that investigates incoherence in arguments. This research is quite different from investigations on text coherence, which is probably more complex on large texts.

2. Construction of a corpus of incoherent requirements

Constructing a corpus of incoherent requirements is very challenging, it is difficult (1) to get real and substantial specifications from companies that have been validated by experts and (2) to extract pairs of incoherent requirements which may appear in remote sections or chapters of a specification.

Incoherence between two requirements may be partial: they may include divergences without being completely logically opposed. Next, incoherence may be visible linguistically, or may require domain knowledge and inferences to be detected and characterized. We focus in this research on incoherences which can be detected from a linguistic and general semantic analysis: these incoherences are domain and style independent and therefore mining them is simpler and probably re-usable over domains. Finally, we focus on incoherences between pairs of arguments, leaving aside incoherences which may arise from larger sets of requirements,

Table 1
Overall distribution of requirements in corpus.

doc ID	nb of pages	nb of requirements	language	domain
1	161	3711	Fr	energy
2	180	3539	Eng	aeronautics
3	152	3498	Eng	aeronautics
4	165	3651	Eng	space
5	198	3833	Eng	transportation regulations
6	193	3951	Eng	space
7	155	3750	Fr	telecommunications
Total	12004	25933	Fr: 7461 Eng: 18472	

e.g. a requirement that is incoherent w.r.t. a set of other requirements. It would obviously be interesting and definitely useful to consider such complex configurations but this involves more language complexity, e.g. the taking into account of discourse, co-text, titles, etc. A gradual approach to such complex forms, based on use-cases will probably be developed in the future. Finding such incoherence situations between sets of requirements is even more challenging than searching for incoherent pairs of arguments.

Our analysis of incoherence is based on a corpus of requirements coming from five companies in five different critical industrial sectors: energy, aeronautics, space, transportation regulations and telecommunications. Companies have requested to be anonymous; named entities (e.g. equipment and process names) in these texts have been replaced by meaningless constants. Specification documents need to be relatively long (above 100 pages) to allow the detection of various forms of incoherences. Shorter and self-contained documents are indeed easier to manually control and have much less incoherence problems. Most specification documents, even for simple equipment are in general very long: incoherences cannot be checked manually, this motivates our project.

The documents in the corpus are in French or in English. The use of two languages is motivated by three factors:

- the difficulty to get real-life corpus of a sufficient size,
- the possibility to develop a system in two languages, since more than half of the specifications are developed in French in France and in French speaking countries,
- the possibility to check if the use of a foreign language (English) by French authors induces additional incoherence problems. Positive results have been obtained within the Lelie project concerning the difficulty for authors to accurately follow authoring guidelines when they write in a foreign language [9].

Our corpus is composed of a total of 1200 pages extracted from 7 documents, where only the requirement parts of these documents have been kept. In particular, introductions, summaries, contexts and definitions, but also diagrams and formulae are not taken into account: they may raise different types of problems. The characteristics of each corpus is summarized in Table 1.

The main features considered to validate our corpus are the following:

1. requirements correspond to various professional activities, and have been authored by technical writers of different professions, over a relatively long time span (about one to two years),
2. requirements correspond to different conceptual levels, from abstract considerations to technical specifications,
3. requirements have been validated and are judged to be in a relatively 'final' state,
4. requirements follow various kinds of authoring norms imposed by companies, including predefined patterns (boilerplates),
5. the documents from which requirements are extracted are well-structured and of different levels of language and conceptual complexity so that incoherences may arise.

To summarize Table 1, the requirement part of each document is between 150 and 200 pages long, i.e. between 3500 and 4000 requirements in each document, resulting in total of about 26000 requirements over all the documents. This corpus size seems to be sufficient to carry out this first analysis. Most documents do not accurately follow the norms for writing requirements, nor do they follow the guidelines imposed by their company. In the next sub-section, the incoherence rate of each document in this corpus is investigated.

3. Extraction of incoherent requirements

It is almost impossible to manually extract incoherent arguments over large texts since this means memorizing with great precision a very large number of requirements. Specification authors can intuitively, via their knowledge of the domain, identify a very limited number of incoherences when they proofread texts, but they know that there are more incoherences, and that these may have important consequences on the process or product being designed.

3.1. Characterizing incoherent requirements

The hypothesis we consider to mine incoherent requirements is that they deal with the same precise topic, but they differ on some significant element(s) which leads to the incoherence. Since requirements should a priori follow strict authoring guidelines (no synonyms, limited syntactic forms), dealing with the same precise topic means that two requirements which are potentially incoherent have a relatively similar syntactic structure and share a large number of similar words, but differ on a few words. This can be illustrated by the two following requirements found in two different chapters of a specification, concerning the same software:

REQ12-7 A minimum of 10 simultaneous requests must be allowed.
REQ34-5 At least 20 simultaneous requests must be allowed.

Our hypothesis states a necessary condition, but not a sufficient condition. Indeed it is possible to have such similarities and differences between two requirements which complement each other or which deal with two close facets of a process or product. If we consider again REQ12-7, then REQ′14-5 is perfectly coherent with REQ12-7:

REQ12-7 A minimum of 10 simultaneous requests must be allowed.
REQ′14-5 A maximum of 20 simultaneous requests must be allowed.

This observation means that the notion of difference between requirements requires an accurate analysis to avoid any erroneous diagnosis.

3.2. Mining for potentially incoherent arguments

Given the corpora defined in section 2, the challenge is to extract a sample of incoherent requirements so that their form and typology can be identified. Since it is not possible to manually extract pairs of incoherent arguments, as indicated in sections 1 and 2, it is necessary to develop tools that extract requirement pairs according to the hypothesis given in 3.1. are only used for corpus exploration, not for automatically detecting incoherent requirements.

The global process to mine potentially incoherent arguments is the following:

1. In a specification, identify requirements among other types of statements such as definitions, illustrations, summaries. Requirements are often identified by a number. If this is not the case, a grammar is used that identifies requirements based on a few specific clues such as the use of modals [12].
2. Mine pairs of requirements which are potentially incoherent based on two metrics developed below:
   (a) a similarity metric, since incoherent requirements deal with the same precise topic and therefore have very close linguistic contents, and
   (b) a dissimilarity metric to identify differences between two requirements.
3. Process the discourse structure of mined requirements. This is realized via the TextCoop discourse processing platform [20]. Of much interest are conditionals, circumstances, goals, purposes, illustrations and restatements. These structures are tagged in each requirement. The goal is to facilitate the manual analysis of incoherence by identifying the discourse structures which are adjoined to the requirement kernel and specify its scope and motivations. For example:
   < circumstance > when the temperature is below 2 degrees, < /circumstance >
   the engines must not be switched off
   < purpose > to avoid any icing </purpose > .
   The whole proposition is a requirement, composed of two discourse structures: the circumstance structure sets constraints under which the requirement is applicable, while the purpose explains the aim of this requirement, as a kind of warning. Some data related to the size of discourse structures compared to the kernel requirement is given in Table 2.
4. Finally, manually inspect the results to identify, for each requirement pair that has been mined, if they are incoherent or not.

Since, at this stage, the form of incoherent requirement pairs is unknown, we developed two metrics with general purpose

Table 2
Overall size of requirements.

doc ID	nb of requirements	average size of kernel in words	average size of discourse structures in words	domain
1	3711	14	18	energy
2	3539	15	21	aeronautics
3	3498	14	23	aeronautics
4	3651	16	16	space
5	3833	13	18	transport regulations
6	3951	15	18	space
7	3750	13	12	telecommunications

Table 4
Summary of incoherence mining with the two metrics.

	total number selected	rate w.r.t. total nb of requirements
Similarity metric	1208	0.46%
Dissimilarity metric	206	0.08%
Really incoherent	87	0.032%

constraints in order to mine a large diversity of situations. The advantage of this strategy is to allow the exploration of potential forms of incoherence on a large scale of possibilities, even if quite a lot of requirement pairs are not inconsistent and must be manually discarded. 1208 pairs of potentially incoherent arguments have been mined following this strategy (Table 4).

3.3. The similarity metric

The similarity metric developed here is quite simple: it is only a tool for corpus exploration. It will be improved before its inclusion into the automatic retrieval of incoherent requirements. The similarity metric relies on structural and linguistic properties of requirements which must follow precise authoring guidelines. The main ones are presented below.

Requirements must be short: in principle they must be smaller than 20 words, but in practice 40 words are frequently encountered. However, a distinction must be made between the requirement kernel which is in general shorter than 20 words, and the adjoined discourse structures, which may be verbose, but are nevertheless necessary for the understanding of the requirement in particular to define its scope. Table 2 summarizes the requirement size distribution by domain and corpus (sizes are rounded up, all types of words are included). Results show that the requirement kernel is about half of the size of the total length of the requirement. Results are relatively similar whatever the domain, telecommunications may be shorter, but this is due to the product being described which is relatively simple.

Language and structure must be standard: For example, from a lexical point of view, synonym terms are not allowed and are filtered or signaled via a domain terminology. The syntactic structure of requirements is very regular and follows general rules of simplified language, for example, passives are not allowed, negation as well as complex coordinations must be avoided, discourse structures are often located at precise places in the sentence, according to predefined templates or boilerplates. For example, a typical requirement kernel is

`subject NP, modal, verb, NP(object), PP*.`

The subject is the actor or the entity being dealt with, PP* indicates a sequence of oblique verb complements such as location, instrument, dative or temporal complements. The order in which oblique verb complements occur in a requirement can be constrained, but in practice this is not frequently followed. For example, for readability reasons, heavy PPs may be moved to the end of the sentence (heavy-NP shift rule). Typical boilerplate are:

`[circumstance, condition, requirement kernel, purpose],`
`[condition, requirement kernel, instrument, manner].`

The discourse structures must be clear and at the right place: e.g. circumstance, condition, purpose, used in the above boilerplates are in general optional, but they frequently occur to specify the scope and the aims of the requirement. They must appear at the right place, as specified in guidelines. Conjunctions of discourse structures of the same type are not allowed: two requirements must be written, even if this obscures the overall comprehension. Boilerplate format is relatively well followed by authors. Our tool can detect incorrect configurations and can possibly suggest reformulations. Although authoring guidelines may slightly be different from one domain to another, an average number of 16 boilerplates are proposed to technical writers per domain.

Lexical choice is constrained: adjectives and adverbs are limited to a few, fuzzy terms are not allowed, useless or verbose expressions must be simplified or avoided, verbs must be limited to the most significant ones and must be precise, etc. For example, in the space domain, the number of verbs used in our corpus is 78 non inflected forms, including auxiliaries, while there are only 37 different adverbs of manner or time.

Considering the figures given above, two requirements with a similar content can be characterized, in a first experiment, by a large set of common words. The nature of the syntactic and discourse structures being very similar or recurrent, these are not taken into account.

Only the words that have topical content are included in the metric, to form the *content requirement signature S*. They are: nouns, proper nouns, numbers, verbs, symbol names, adverbs, and boolean or scalar adjectives. Their identification is based on WordNet resources. The other terms, which mostly play a grammatical role, are not used in the metric, they include: prepositions, determiners, modals, auxiliaries if they are not the main verb, negation, connectors, coordination and pronouns. They play a role in the incoherence analysis, but do not convey the themes of the requirement. These terms are available in the TextCoop platform [20]. In the signature, duplicate words count for a single occurrence, while compound terms are considered as a single term, similarly to the bag-of-words model.

For example, in:

the maximum length of an imaging segment must be 300 km,
the words that are considered in the metric are:

Table 3
Similarity distribution.

Requirement size	total number of requirements in category	similarity rate	number of similar pairs
6 to 15 words	3794	90%	84
		85%	185
		80%	408
		75%	564
16 to 25 words	6563	85%	117
		80%	156
		75%	423
		70%	788
26 to 35 words	9649	85%	328
		80%	437
		75%	732
		70%	951
36 to 45 words	3365	85%	97
		80%	229
		75%	345
		70%	651
> 46 words	2562	80%	187
		75%	201
		70%	984
		65%	1297

maximum, length, imaging, segment, be, 300, km.

These form the *content requirement signature* of that requirement, composed of 7 terms.

To evaluate the similarity between two requirements R_1 and R_2, the similarity metric considers the *content requirement signatures* S_1 and S_2 of R_1 and R_2 and computes their intersection I and union U, considering the noninflected forms of words. The similarity rate is:

$$card(I) \;/\; card(U);$$

where `card` is a function that computes the number of words in a set. The intuition behind this simple metric is that, besides the number of different terms, the number of common terms must be taken into account so that the characteristics of each element of the pair are taken into account since similarity is characterized by a threshold.

Requirements have in general between 6 and 50 words, with an average size of 34 words, following the figures given in Table 2. An average rate of 63% of these words in a requirement is part of the signature.

The next challenge is to define a similarity threshold from the similarity metric. Table 3 shows the distribution of similar requirement pairs depending on their length and similarity rate. For example, the second line considers requirements which are between 15 and 25 words long, and explores the four similarity rates: 85, 80, 75 and 70%. This table shows that results are quite different depending on the requirement length. An inflection is observed, for example on line 2 below 80%. We therefore consider that 80% is a good threshold for our experiment, below this rate the risk is to have too many pairs which are not strongly similar and with little risk of real incoherence. To conclude from this experiment, thresholds are defined as follows:

1. between a total of 6 and 15 words long, the similarity rate is greater or equal to 85%,
2. between a total of 16 and 35 words long, the similarity rate is greater or equal to 80%,
3. above a total of 36 words long, similarity rate is greater or equal to 75%.

Long requirements often include peripheral information which means that the threshold for the similarity weighted rate should be lower. This three level classification can still be slightly improved, depending on the authoring style of the requirements, the domain via its terminology and the conceptual complexity, but it seems sufficiently operational as a tool for analyzing a corpus. With these thresholds, a total of 1208 requirements, out of a total of 25933 (0.46%) are potential candidates for incoherence before application of the dissimilarity metric.

A side effect of this analysis is to also mine duplicates or very closely related requirements which could possibly be eliminated. As an illustration, via the similarity metric used alone, with a threshold of 90% independently of the requirement number of words, our corpus contains at least 375 requirements which are very closely related and could be analyzed as duplicates.

3.4. The dissimilarity metric

The dissimilarity metric is based on the presence of terms which are conceptually different in a pair of requirements. It parallels the similarity metric: it searches for terms which introduce a clear difference between two requirements. Differences which introduce

forms of incoherence which have a linguistic basis frequently correspond to forms of antonyms [4] in lexical semantics. Differences may occur in the kernel requirement or in discourse structures which introduce restrictions or expand the kernel: conditions, circumstance, elaboration, illustration. Discourse structures which develop purposes are not considered in this analysis because they are more at the periphery: they act as a support, they do not specify a constraint. Finally, our observations show that differences are essentially located in the verb complement part, not in the subject because it sets the context of the requirement (e.g. ground taxi speed must be …). However this is not a general rule, but this observation can be used as a heuristics to rank incoherences.

From the analysis of pairs selected by the similarity metric, differences which may lead to incoherences can be characterized by the main following situations:

- **use of different values in a similar construction** in the two requirements. For our manual analysis we estimate that a difference rate must be of at least 10% between the two values to be significant. For example, in the domain of airport regulations:
  ground speed during taxi must be below 20 kts *to minimize fuel consumption* vs.
  ground speed during taxi must be 25 kts *to comply with ground traffic regulations.*
  These two requirements do not concern the same constraint: they are not therefore totally incoherent. There is however a clear discrepancy which introduces a partial incoherence. These two requirements are similar, in spite of their purpose that is different.
- **use of different arithmetical operators or boolean constraints**, e.g.:
  In S, the probe X30 shall be preheated up to 30 degrees before doing E.
  the probe X30 in S shall be preheated to 50 degrees to do E.
- **use of different named entities**, including unit names and their acronyms:
  V1 is 135 kts with no head wind and gusts vs.
  V1 is 136 N. miles without any head wind and gusts.
  These two requirements are coherent, the units which are used are identical, but the name must be unique. This is not the case in the next example, with the use of two different color names:
  TCAS alarm message must appear in red on the FMGC screen, vs.
  TCAS alarm message must appear in purple on the FMGC screen.
- **use of antonym prepositions or connectors**, e.g.: before/after, near/on, etc. Discrepancies in preposition compounds or lack of PP (preposition vs. none) is also a source of incoherence:
  Engines must be switched off upon arrival at the gate, vs.
  Engines must be switched off at the gate.
  This latter requirement describes a state and omits the synchronization introduced in the preposition compound 'upon arrival at' formed of an adverbial combined with a preposition.
- **use of contrasted modals:** e.g.: *must/recommended*, e.g.:
  Air conditioning must be switched off when doors are open, vs.
  is is recommended to switch off air conditioning when doors are open.
- **presence of typical structures such as manner or temporal adverbs in a requirement and not in the other**, e.g.
  the main pipe pressure must be carefully controlled every hour by a certified agent, vs.
  the main pipe pressure must be controlled every hour by a certified agent.
  Although the manner adverb 'carefully' is fuzzy and should be banned from requirements, it nevertheless warns the agent of the necessity of doing the control accurately. In that case the inconsistency must be resolved and the adverb of manner made more explicit.
- **presence of two main clauses (two events) in one requirement and a single main clause in the other** (single event), which may indicate that an event has been omitted or that the two events have been merged.
- **modification of the hierarchy between two events**, e.g.
  The operator must open the pipe in 'full' position and check the oil pressure, vs.
  The operator must open the pipe while checking the oil pressure.
  In the second requirement, the two events occur simultaneously, whereas in the first, they should, a priori, follow each other.
- **presence of a conditional or a circumstance in one of the requirements and not in the other one**. This changes the scope of the requirement.

Provided that two requirements have been ranked as similar by the similarity metric, the detection of at least one of these criteria in a pair of requirements entails that they may be incoherent. More than one criteria reinforces the risk of incoherence, however, there is no dissimilarity measure at this level.

3.5. Corpus analysis results

The similarity and then the dissimilarity metrics have been used on the seven texts of the corpus separately since they each text deals with a different topic. The results for these seven texts can be summarized as follows. 1208 requirement pairs have been characterized as similar (0.46%). Then, via the application of the dissimilarity metric, a total of 206 pairs of requirements have been characterized as potentially incoherent (i.e. about 0.08% of the total number of requirements).

From this set of pairs which are potentially incoherent, via a manual analysis, 87 requirement pairs (on average 42%) indeed

show some form of incoherence. This is not a very high number of incoherent arguments: this confirms expert intuitions that incoherences are sparse, but there are probably more incoherences, including those which involve domain knowledge instead of just linguistic analysis alone. Even if the number of incoherences that have been found so far is small, it is nevertheless crucial to take them into account. Requirement authors admit that consequences may indeed be high if they are neglected. Finally, our system needs to be improved. Our own informal evaluation is that at the moment we detect about one third of the total number of incoherences.

Seven pairs of requirements selected by the similarity metric which are incoherent (manual analysis) have not been recognized as potentially incoherent by the dissimilarity metric. The dissimilarity metric seems therefore to be quite efficient with a silence rate of 8%. Finally, to illustrate the problem of incoherence which occur in non-adjacent pairs of requirements, 21 pairs have been found in the same text section (i.e. among between about 80 and 150 requirements), while 66 pairs are more distant in the text and would probably not have been identified without our tool.

These results are summarized in Table 4. Even if it is small, this sample of 87 requirement pairs allows us to develop an analysis of incoherence among requirements, and then to develop relatively generic templates that go beyond these observations. These should be able to mine a higher number of incoherent requirement pairs (an increase of 44% is reported in section 6.5). Finally, the rate of 0.032% of incoherent requirement pairs is certainly largely underestimated, since only linguistic data is used. Nevertheless, this means about 12–13 incoherent requirement pairs on average in each of our 7 documents (with an increase to 20 pairs when using the patterns already implemented), which is not negligible and may entail various types of risks and misconceptions.

4. Annotating incoherence in requirements

The next stage of our analysis is to identify more precisely the various forms incoherence has in requirements from the 87 pairs that have been mined. For that purpose, we defined a set of annotations. The discourse processing used to mine requirement pairs provides a first level of annotations as illustrated in section 3.2. In case of errors in the discourse analysis, in particular in identifying the boundaries of the discourse structure, some manual tuning can be done. Since this is an analysis step to develop mining patterns (section 6), annotations are made manually by the author and a student, some confirmation was given by requirement authors, but their availability is often very limited. Agreement is around 80%, it is in fact difficult to precisely measure agreement when some attributes are informal. A final annotated corpus was developed on which annotators agree. It will be used for the results presented below.

The annotations characterize the string of words which differ between the elements of a pair and the nature of the difference(s). This task allows to categorize errors (section 5) and to define templates to mine incoherent arguments in specifications (section 6). Incoherences are specified informally by two attributes of the main tag < incoherence > . These attributes informally specify in natural language the type of the incoherence (tag 'type') which will be used to develop a categorization and characterizes, via the strings of words involved and an operator, the nature of the incoherence (tag 'incoherence'). Word differences between two requirements are identified between < diff > tags, < req > identifies a requirement, and < event > tags the events in requirements which contain two or more explicit events.

Here are a few examples that illustrate the method and the difficulties. Examples 1 and 2 are relatively straightforward whereas examples 3 and 4 are much more complex to annotate and to characterize. Each example includes a pair of requirement represented between two < req > tags:

Example 1.

< incoherence type = "numerical variation" incoherence = "300 ≠ 100" >
< req > the maximum length of an imaging segment is < diff > 300 < /diff > km. < /req >
< req > the maximum length of an imaging segment is < diff > 100 < /diff > km. < /req >
< /incoherence >

Example 2.

< incoherence type = "arithmetical expression divergence" incoherence = "(< 30) ≠ 50" >
< req > In S, the probe X30 shall be preheated < diff > up to 30 < /diff > degrees < circumstance > < diff > before doing < /diff > E. < /circumstance > < /req >
< req > the probe X30 in S shall be preheated < diff > to 50 < /diff > degrees < purpose > < diff > to do < /diff > E. < /purpose > < /req > < /incoherence >

In this example, while the arithmetical expression introduces the incoherence, the distinction between circumstance (first requirement in the pair) and purpose (second element of the pair) must be made flexible so that the synonymy between 'doing' and 'to realize' can be identified.

Example 3. English gloss of French example: *at flight level 30 in normal flight conditions, the maximal 20 degree flap extension speed is 185* kts versus:
at flight level 30 in normal flight conditions, extend flaps to 20 degrees and then reduce speed below 185 kts*:*

< incoherence type = "event synchronization with mismatches"

incoherence = “(< 185 kts) ≠ unknown ” >
< req > < circumstance > A FL 30 ASL et dans les conditions normales < /circumstance > , la vitesse < diff > maximale de < /diff > sortie des volets à 20 degrés < diff > est < /diff > 185 kts. < /req >
< req > < circumstance > A FL 30 ASL et dans les conditions normales < /circumstance > , < event > sortir les volets à 20 degrés < /event > puis < event > < diff > réduire la vitesse en dessous de < /diff > 185 kts. < /event > < /req >
< /incoherence >

In this example, a general rule (first requirement) is contrasted with a requirement that describes a procedure composed of two instructions. In the second requirement, the temporal connector 'puis' (then) suggests that the constraint stated in the first requirement is split into two consecutive parts, with the risk that the constraint is not met as it should, e.g. flaps are extended at an unknown speed, and then the speed is reduced to 185 kts.

Example 4.

< incoherence type = “incompatibility between events”
incoherence = “(stop every 24 h) ≠ (no stop during the entire update) “ >
< req > the system S administrator must make sure < diff > to stop the system S every 24 h < purpose > for maintenance. < /purpose > < /diff > < /req >
< req > the system S administrator must make sure < diff > that the update of system S database is not interrupted. < /diff > < /req >
< /incoherence > .

In this example, a potential incoherence may arise if the system is stopped for maintenance while its associated database is being updated. The terms ‘stop’ and ‘not interrupted’ are opposed: they induce the incoherence.

These annotations have been developed to explore the incoherence problem in requirements, they remain informal, in particular for the 'type' attribute. Incoherences are indeed very diverse and do not lend themselves easily to a simple characterization. The goal is to have a first level of analysis, before implementing templates that would mine incoherent pairs of requirements.

5. A preliminary categorization of incoherence in requirements

From our sample of 87 incoherent requirement pairs, a preliminary categorization of incoherence types can be elaborated on a linguistic and conceptual basis. This categorization shows that incoherence has multiple facets. Some pairs that have been mined may be clearly incoherent whereas others need a domain expertise. Incoherence may be clear for some requirement pairs while in other cases incoherence is subtle, partial or shallow. Therefore, the severity level of an incoherence is difficult to measure.

The categorization presented below is defined empirically from our corpus and is based on simple linguistic considerations. This categorization leads to the definition of templates and lexical resources to mine pairs of incoherent requirements (section 6). The examples below are direct corpus samples (translated into English for the French examples). They are numbered Ri a/b where i numbers examples and a and b are the two elements of an incoherent pair.

5.1. Partial or total incompatibilities between expressions

The most direct and simple type of incoherence is composed of ‘local’ differences. These differences are numerical, related to logical or arithmetical expressions, expression of negation, or come from lexical semantics antonyms.

5.1.1. Incompatible numerical values or arithmetical expressions

After some discussions with requirement authors, although this remains debatable, it turns out that a numerical variation becomes problematic in a pair of close requirements when a difference of more than 10% is observed:

R1a- Make sure to preheat the probe X30 to a maximum of 30 degrees.
R1b- The probe X30 must be preheated to at least 50 degrees.

The computation of the difference threshold related to the 10% difference is given in section 6.4, item C. Here 10% difference means a maximal difference of 4 between the two values.

While R1 is clearly an incoherence in the evaluation of the temperature, R′1 is probably a typo which entails an incoherence:

R′1a- the maximum length of an imaging segment is 300 km.
R′1b- the minimum length of an imaging segment is 30 km.

Numerical differences may be due to different contexts, for example a probe may be preheated to a certain temperature for a given operation and to another temperature for another operation.

5.1.2. Incompatible temporal organization between events

The use of temporal marks which are antonyms to a certain degree is also an important factor of incoherence. For example, in:

R2a- Update data marking when transferring data.
R2b- Update data marking before transferring data,

the contrast between 'when' (overlap) and before (precedence) is a form of incoherence, provided that 'before' is a strict precedence relation.

5.1.3. Incompatible modal variation

The use of modals in requirements is very precise: it indicates their criticity level. Given a priority or criticity level for a requirement, the choice of the appropriate modal is defined in general in authoring guidelines. Therefore, modal variation involving a different critical level is a specific but important source of incoherence:

R3a- it is recommended to stop the engine before E.
R3b- the engine shall be stopped before E.

may entail a risk, a malfunction or an accident.

5.1.4. Incompatible uses of adverbs

The use of adverbs in requirements is rather limited. Guidelines frequently give or suggest lists of adverbs which can be unambiguously used. These are essentially manner and temporal adverbs, even if some of them are fuzzy. Adverbial incompatibilities or discrepancies involve pairs of direct antonyms: *slowly/quickly*, but also pairs which are related via inference, such as: *carefully/quickly*, but which nevertheless suggest antonymy: carefully requires time which 'quickly' does not presupposes, indeed, *carefully and quickly* although possible sounds odd in a conjunction.

5.1.5. Incompatible forms of quantification

Although also quite restricted in requirements, some forms of quantification over objects is often necessary. Some specific quantification aspects which lead to incoherences such as the difference between the quantifiers *every/each, most/all* has been observed. The first pair concerns a difference in distributivity whereas the second pair of quantifiers is quantitative.

5.1.6. Incompatible forms in PPs

Incompatibility between prepositional phrases can be characterized in a pair of requirements by the use of different temporal, spatial or instrumental restrictions in verb complements:

R4a- Service D must be available only from the screen.
R4b- Service D must be available in any configuration.

A domain ontology is necessary to detect most of these incompatibilities. However, in this example, the quantifier contrast between 'only' and 'any' and the fact that a different term is used in the VP complement suggests a risk of incompatibility. This is particularly the case in technical writing where synonyms are not allowed.

Among these various forms of incoherences, the three first cases are the most frequent. Our corpus includes 34 cases that fall in this category, among which 25 belong to the three first cases.

5.2. Incompatible events

In this category fall pairs of requirements that describe incompatible events the detection of which often requires domain knowledge. In spite of this knowledge limitation, a number of differences appear at a 'surface' level and can be detected for a large part on a linguistic basis. Examples 3 and 4 in section 4 are typical incoherent pairs of events.

The first illustration below relates a type of incoherence that is difficult to predict without the presence of the antonym *stop/not interrupted*. This difference suggests a closer look at these two close requirement:

R5a- The system S must be stopped every 24 h *for maintenance.* versus
R5b- The update of the database via the system S must not be interrupted.

Next, the presence of two events in an argument pair whereas the other one has only one event requires a close control, in particular when a temporal connector 'then' is used that suggests precedence:

R6a- the maximum 20 degree flap extension speed is 185 kts. versus
R6b- extend flaps to 20 degrees and then slow down to 185 kts.

This is a business error that any domain expert should typically be able to avoid. Note that R6a is a general rule which should be preferred to R6b which is a procedural statement, not a requirement.

5.3. Contextual incompatibilities

This category includes various types of discourse structures (e.g. condition, circumstance, goal, illustration, etc.) associated with the main body of two similar requirements which could induce forms of incompatibility:

R7a- during the project specification phase, both on-line and off-line access must be available.
R7b- during the project implementation, both on-line and off-line access must be available.
R8a- a rule includes facts and their criticity.
R8b- a rule includes facts and a description.

R7 illustrates a case where the context (circumstance) is different (*specification phase* versus *implementation*) but expects the same behavior. One may expect a negation in one of the two main propositions 'must not be available' to motivate these two requirements. If these two requirements are coherent for an expert, then it is preferable to merge the two circumstances to avoid any doubt.

R8 shows two enumerations which are different and should probably be adjusted. R8 is a frequent case where enumerations are either incomplete (not recommended in requirement guidelines), or manipulate objects at different levels of abstraction: the term 'description' could include the description of the 'criticity'.

5.4. Terminological variations and other discrepancies

In this category, requirements which largely overlap are considered. Their slight differences may be symptomatic of a partial inconsistency. These cases are relatively frequent and typical of documents produced either by several authors or over a long time span (where e.g. equipment names or properties may have changed). Typical examples are:

R8a- the up-link frequency, from earth to space, must be in the s band.
R8b- the down-link frequency, from space to earth, must be in the s band.
R9a- those tests aim at checking the rf compatibility of the ground stations used during the mission and the tm/tc equipment on board.
R9b- those tests aim at checking the rf compatibility of the ground stations used during the mission and the on board equipment.

In R8 the use of the s band in both situations requires a control by an expert, in R9 *tm/tc* must be compared with an on board equipment.

R10 is somewhat ambiguous and contains implicit knowledge which makes the identification of the incoherence slightly more challenging:

R10a- the upper attachment limit must not exceed 25 GB.
R10b- It must be possible to specify a maximum limit for the storage capacity of an attachment.

Indeed, either the limit is set to 25 GB or it can be any value to be specified, but R10b may be understood as saying that it can be specified but always below 25 GB if R10a is correct, which needs some expert control.

5.5. Category synthesis

The distribution observed on our corpus of 87 requirement pairs is given in Table 5. The results presented in this table remain indicative because of the small size of this corpus. They also correspond to incoherences which can be detected only on the basis of linguistic analysis.

Table 5 shows that the different categories of incoherence are non-negligible. Probably category 2 is larger in the reality since it does not necessarily have typical language clues.

Most of the incoherences in categories 1 and 4 can be detected on the basis of 'local' content analysis paired with linguistic analysis and general purpose lexical semantics data, in particular antonyms e.g. for prepositions, temporal connectors, arithmetical

Table 5
Distribution of forms of incoherence.

Category	Number of occurrences and rate w.r.t. total number
(1) incompatibilities between expressions	34, about 40%
(2) incompatible events	12, about 13%
(3) contextual incompatibilities	28, about 33%
(4) variations between requirements	13, about 14%

operators. Category 3 requires more domain knowledge, while category 2 is more challenging. The next section develops patterns for automatically detecting the simplest forms of incoherences described in this section.

6. Automatic analysis of incoherences

The main challenge is now the definition of templates to mine incoherent pairs of requirements, with their associated linguistic resources. This section shows how templates have been developed for the categories 1 and 4 described in section 5. Categories 2 and 3 are more complex and will be treated in a second stage. The similarity and dissimilarity metrics have been revised and improved from the experiences reported in section 5. Improved metrics combined with strict incoherence patterns should guarantee a higher accuracy rate than in the manual corpus analysis.

The templates presented below are implemented in < TextCoop > a platform we defined for discourse analysis that allows an easy declarative specification of discourse templates.

6.1. The global processing strategy

The global strategy allows a comprehensive analysis of a requirement document. The complexity in terms of processing time is quite high since requirements must be compared by pair. Coherent requirements are stored, in their occurrence order, one after the other in a requirement database to allow for an easy coherence control. When a new requirement is read from a text, it is compared with all the requirements in that database.

The processing strategy is organized as follows:

1. a new requirement R_i is read from the source text, its discourse structure is tagged, including the kernel requirement portion,
2. R_i is then compared to all the elements R_j already stored in the requirement database. For that purpose:
   (a) given, a requirement R_j in the database, the revised similarity and dissimilarity metrics are activated to check whether R_i and R_j may potentially be incoherent,
   (b) If this is the case, then incoherence patterns are activated to detect potential incoherences. If this is the case, a potential incoherence warning is produced and R_i is not added to the database,
   (c) if no incoherences have been detected, then R_i is added to the incoherence database.

In the subsections that follows, we present the revised metrics, < TextCoop > , the discourse analysis platform on which the incoherence analysis runs, Dislog, the language used to implement patterns, and then the different incoherence patterns that have been defined at this stage.

6.2. The similarity metric revised

The similarity metric clearly needs to be refined: a simple word to word match and count is obviously not accurate enough, even if it allowed us to investigate the incoherence problem from scratch. In this first revision, to be evaluated, the similarity metric is extended to the taking into account of the morphology of nouns, adjectives and verbs, realized by the morphological analyzer integrated into < TextCoop > , which is relatively simple for English. A few lexical variations are also introduced, in particular the use of more generic terms, on the basis of the domain ontology when it exists. Next, a few terms are no longer considered in the similarity measure: in particular terms which express stress (e.g. *necessary, crucial*), which are author dependent and not central to a requirement. Finally, the contents of purpose clauses and warnings are no longer considered because they are not part of the requirement, they simply justify it. Circumstances and conditions are obviously kept in the metric.

Several harder problems such as pronominal references, the use of more generic terms, implicit terms, and ellipsis must be taken into account in some way in future versions if the current one turns out not to be accurate enough. As suggested by the two reviewers, exploring the possibilities offered by the Cosine similarity metric would be of much interest since it is used in high-dimensional positive spaces, which is the case here. This metric could be used in a more advanced implementation of the system once all the linguistic categories have been identified.

At this stage, we feel that the dissimilarity metric is accurate enough since it has a low silence rate of about 8% (section 3.5), as evaluated by the two same annotators (author and a student, and some confirmation from technical authors). However, it seems that the dissimilarity metric is the most decisive one, it would be of much interest to analyze in detail the nature of the dissimilarity.

Two requirements R_i and R_j processed by the similarity and dissimilarity metrics which are potential candidates for the incoherence test (section 6.4) are tagged as shown in section 4, without the attributes of the < incoherence > tag which were design for manual analysis purposes. Only the circumstance, illustration and condition discourse structures are included since they contribute to the incoherence analysis. In Example 1, R_i is the requirement read in the text while R_j a requirement in the database. They have been detected to be potentially incoherent. They are tagged as follows including the circumstance discourse structure:

Example 1a.

< potential_incoherence >
< req > < circumstance > in normal weather conditions < / circumstance > , the maximum length of an imaging segment

is < diff > 300 < /diff > km. < /req >
< req > < circumstance > in standard weather conditions < / circumstance > , the maximum length of an imaging segment is < diff > 100 < /diff > km. < /req >
< /potential_incoherence >

The requirement kernel is the segment that is not tagged by a discourse structure. In this example, *standard* and *normal* are synonyms and therefore no longer appear between difference tags.

6.3. An introduction to < TextCoop >

Most of the process of identifying incoherent requirements is implemented in TextCoop, which offers a simple and declarative framework, well-suited for our needs. In this section, the main features of TextCoop which are important for this project are presented. TextCoop is presented in detail in Ref. [21].

TextCoop is a logic-based platform designed to describe and implement discourse structures and related constraints via an authoring tool. Dislog (Discourse in Logic) is the language designed for writing rules and lexical data. Dislog extends the formalism of Definite Clause Grammars to discourse processing and allows the integration of knowledge and inferences. Dislog also extends the possibilities offered by regular expressions. TextCoop is currently used to process various kinds of procedural texts, industrial requirements and regulations, news texts and didactic texts. It is used in projects dedicated to health and ecology safety analysis in industrial procedures (the Lelie project) and in opinion analysis, in particular for argument extraction. TextCoop is in a relatively early stage of development, it offers different functions than well-known platforms such as Gate or Linguastream, such as the inclusion of reasoning procedures during the discourse analysis process.

< TextCoop > is implemented in Prolog SWI as a meta-interpreter. The implementation is relatively efficient. In average, about 10 Megabytes of text are processed per hour on a standard PC with a set of 60 rules or patterns and 1000 lexical items.

6.3.1. The dislog language

Rules basically recognize most basic discourse units, including forms such as requirements.

A rule in Dislog has the following form:

L(Representation) → R, {P}. where:

1. L is a non-terminal symbol.
2. Representation is an XML structure with attributes, or a (partial) dependency structure. It may also contain messages when e.g. incoherences have been detected.
3. R is a set of symbols as described below, and
4. P is a set of predicates and functions implemented in Prolog to deal with various forms of reasoning (e.g. to check for types based on ontological knowledge, etc.).

R is a finite sequence of the following main elements: terminal symbols that represent words, expressions, punctuations, various existing html or XML tags, preterminal symbols which can be associated with a type feature structure, non-terminal symbols implement 'local grammars', i.e. grammars that encode specific syntactic constructions, optionality and iterativity marks over non-terminal and preterminal symbols, as in regular expressions.

A major element is represented by gaps, which are symbols that stand for a finite sequence of words of no present interest for the rule which must be skipped. A gap can appear only between terminal, preterminal or non-terminal symbols. Dislog offers the possibility to specify in a gap a list of elements which must not be skipped.

For example, a simple pattern to recognize a condition is, given in readable form:

```
[cond-expr, gap(X), comma].
cond-expr-> if/whether.
comma-> ','.
```

A cluster of Dislog rules or patterns is in general needed to recognize a given basic discourse unit (e.g. requirement kernel, illustration, purpose) since they often have different surface forms. In (Saint-Dizier 2014) the accuracy of the system is analyzed. For requirements, an accuracy of about 90% is reached to recognized the structure of requirements and of the discourse structures advocated in this article.

Dislog offers selective binding rules to bind structures under constraints. Rule execution is managed by rule concurrency management.

6.3.2. The < TextCoop > environment

The < TextCoop > environment includes authoring tools for rules and a number of useful lexical resources. The following resources have been designed for French and English:

- lists of connectors, organized by general types: time, cause, concession, etc.
- list of specific terms which can appear in a number of discourse functions, e.g.: terms specific of requirements (modals),

illustration, summarization, reformulation, etc.
- lists of verbs organized by semantic classes, close to those found in WordNet, that we have adapted or refined for discourse analysis,
- list of terms with positive or negative polarity, list of intensifiers or downtoners (e.g. to measure emphasis or stress in opinion analysis),
- local grammars for e.g.: temporal expressions, expression of quantity, etc.
- some already defined clusters of discourse function rules to recognize general purpose discourse functions such as requirements, illustration, definition, reformulation, purpose, circumstance and condition.
- some predefined functions and predicates to access knowledge and control features (e.g. subsumption),
- utilities for integrating knowledge (e.g. ontologies) into the environment and in Dislog rules.

6.4. Definition of incoherence patterns

In this section we show how incoherence patterns are defined. We concentrate on categories 1 and 4 presented in sections 5.1 and 5.4 above. These are the simplest categories in terms of processing and linguistic resources. Category 2 requires an in-depth analysis of event structures while category 3 needs a domain terminology. This latter category is not more complex than categories 1 and 4, but it needs some domain resources which are not necessarily accessible.

6.4.1. Extraction of the differences between two requirements

Given two requirements which are potentially incoherent, represented as illustrated by Example 1a, a Dislog rule extracts the difference sequences for each requirement. The result is represented as a list of lists, where the each list contains the set of differences:

[[diff in req1], [diff in req2]]

for Example 1a, this list is simple since there is only a difference sequence:

[[300], [100]].

In Example 2 in section 4, there are two sequences for each requirement:

[[[up,to,30],[before,doing]], [[to,50],[to,do]]].

In our corpus, it turns out that potentially incoherent requirements have either one or two sequences of words with differences, as illustrated above. A Dislog rule to extract a single sequence is written as follows, in readable form, where symbols and terminal elements are separated by commas:

```
[ < potential incoherence > ,
 < req > , gap(X), < diff > , Seq(A), < /diff > gap(Y),
 < req > , gap(Z), < diff > , Seq(B), < /diff > , gap(T)]
– > [[Seq(A)], [Seq(B)]].
```

gaps skip finite sequences of words of no present interest; Seq(A) and Seq(B) extract sequences of words in the difference zone. The result is these two sequences, stored in a list of lists.

6.4.2. The lexical resources of incoherence

The first step is the elaboration of the lexical and morphological resources which are needed. Incoherence is for a large part based on antonyms. Cruse (1986) gives a large survey of the different types of antonyms and how they can be described. Most of his observations have been implemented in WordNet.

The lexicon of grammatical terms (e.g. prepositions, connectors, determiners) and of adjectives and adverbs (manner and temporal) is not very large in requirements due to authoring constraints. In terms of antonyms, the resources already available in the TextCoop platform include a relatively comprehensive lexicon partly based on WordNet, which is not so accurate concerning adjective and adverb antonyms, and partly constructed manually. These were initially developed for opinion analysis, but turn out to be appropriate and accurate enough for analyzing requirements. Besides the lexical aspects, the morphological analyzer provided in the TextCoop platform is also used.

In our corpus, three types of antonyms characterize incoherence. In the lexicon of our implementation, this is realized by the following structures, with lists of terms oriented from negative to positive:

1. boolean antonyms where there is no alternative between the two terms, typically: (*dead, alive*) or (*on, off*), other pairs of terms may include lexical variants with possible *trans*-categorical realizations, e.g.: (*maximum, under*), (*minimum, at least*), (*only, any*), these two pairs are analyzed as antonyms,
2. antonyms involving triples, where the second element denotes a neutral position. This element may not be linguistically realized but has a conceptual reality:
   (*before, in parallel, after*), (*cautiously, _, quickly*),
3. non-branching proportional series which structure terms according to a given dimension, e.g.:
   contact: (far, near, on),
   temperature: (frozen, cold, mild, warm, hot),
   in these series there is frequently a kind of neutral term which occurs approximately in the middle of the series. Terms on each

side of the series induce various degrees of incoherences, depending, e.g., on how remote from each other they are on the scale.

Besides antonyms, although the use of synonyms is not allowed in requirements, some general purpose terms may have quasi-synonyms, which must not be included in the dissimilarity metric and should not induce any form of incoherence. A few of them are used, they are defined in the < TextCoop > platform lexical resources. For example, terms such as: *(do, doing, realize, carry out)* are quasi-synonyms, possibly with morphological variations (do-doing). In example 2, section 4, *to do* and *doing* do not introduce any incoherence, even if the progressive form induces a different aspectual value.

6.4.3. Incoherence pattern model and implementation

The patterns we have developed so far rely on the lexical resources resented in the previous section. The main patterns are presented here. Additional ones develop exceptions or specific cases.

The patterns related to categories B, C, D and E described in section 5.1 are modeled by the same pattern, with the use of different lexical categories:

Pattern1:

```
incoherence-pattern([[A], [B]]):-
in(A, TermA), in(B, TermB), antonym(TermA, TermB).
```

This pattern non-deterministically extracts TermA in the sequence of the different terms A in the first requirement, and TermB in the sequence of the different terms B, and checks whether they appear as antonyms in the lexicon. `antonym(X,Y)` is a predicate that traverses the different lexical resources presented in the sub-section above. If an antonym is found, then an alert is produced that indicates that the two requirements considered may be incoherent. The alert mentions the type of incoherence: temporal between events, criticity (modal), adverbial (manner or temporal) or quantification. For example, in the case of incoherent adverbs, the incoherence is signaled to the author as follows:

RQ12-3 Update data marking when transferring data.
RQ27-5 Update data marking before transferring data.
⇒ Event incoherence due to a discrepancy between ‘when’ and ‘before’.

In a Word document, this incoherence could appear in a left margin, comparable to the ‘search’ feature.

The category F reported in section 5.1, where two PPs are different, involves the detection of an antonym and a control on the remainder of the PP, namely that they contain terms, including closed categories, which are different:

Pattern2:

```
incoherence-pattern([[A], [B]]):-
in(A, TermA), in(B, TermB), antonym(TermA, TermB),
diffW(A, B).
```

where `diffW(X,Y)` is true if X and Y are different sequences of words. This control may be too systematic, but it avoids to take into account domain knowledge. The evaluation given below confirms the soundness of our approach.

Category A is more complex. Consider again the example:

R1a- Make sure to preheat the probe X30 to a maximum of 30 degrees.
R1b- The probe X30 must be preheated at at least 50 degrees.

The detection of the incoherence involves two parameters:

1. a difference of at least 10% between the values given in the two requirements. If V_1 and V_2 are those values, then the difference threshold between them must be at least: $((V_1 + V_2)/2)*0.1)$. For example, for $V_1 = 30$ and $V_2 = 50$, the difference threshold is 4.
2. the lexical semantics of the mathematical operators, which may be antonyms. If Op_1 is the operator in the first requirement and Op_2 the operator in the second one, then *antonym*(Op_1, Op_2), with the orientation developed in B above, is true if they are antonyms. In the example, we have *antonym*(*maximum*, *at least*) which is evaluated to rue since ‘maximum’ precedes ‘at least’ in an antonym scale.

Then, an incoherence is detected as follows:

- if there is no mathematical operator, then the requirements are incoherent if the difference between V_1 and V_2 is greater than the threshold of 10%,
- if there is an operator, the requirements are incoherent if the orientation of the operators and the values, which differ of at least 10%, must be identical: $Op_1 < Op_2$ and $V_1 < V_2$ or vice versa.

The example above meets the second condition: the two requirements are incoherent.

The pattern is the written as follows, with the two configurations that induce incoherence:

Pattern3a:

```
incoherence-pattern([[A], [B]]):-
Value(A, VA), value(B, VB), diffV(VA,VB), VA < VB,
antonym(TermA, TermB), precedes(TermA, TermB).
```

Pattern3b:

```
incoherence-pattern([[A], [B]]):-
Value(A, VA), value(B, VB), diffV(VA,VB), VA > VB,
antonym(TermA, TermB), precedes(TermB, TermA).
```

In these patterns, the predicate *diffV* (*X*, *Y*) is true is the difference between X and Y is greater than 10%.

We have defined at the moment 4 main patterns and 7 additional ones to deal with specific cases or exceptions.

6.5. Analysis of the results

Since it is not possible to have a reference corpus, manually annotated, that could serve as a test corpus, for the reasons advocated in section 2, the corpora used for the incoherence categorization is re-used. Our goal is to see how the patterns perform for categories 1 and 4. Results are the following:

- initial manual analysis with first version of similarity and dissimilarity metrics (reported in section 3.5 and in Table 4): 47 pairs of incoherent requirements found, average accuracy: 42%,
- automatic analysis with patterns and revised similarity metrics, as reported in this section: a total of 94 pairs potentially incoherent have been mined, out of which 68 are indeed incoherent (manual analysis), therefore, the accuracy rate is 72%, which is much better than the initial result.

Besides the accuracy improvement, it is interesting to note that the implementation has allowed the detection of more incoherence cases: 68 pairs instead of 47, therefore an increase of 44%.

6.6. Discussion

The accuracy rate of 72% is the minimal acceptable accuracy for requirement authors. They however recognize that these results constitute a great help to improve their texts.

The main directions we plan to improve incoherence recognition and, at the same time, to limit noise are characterized by the following situations:

(1) **Different forms for a similar content:** two requirements may deal with the same point using slightly different expression means, e.g. for values. For example, values and units may be different, intervals or arithmetical constraints may be used instead of a single value, but these expressions remain globally equivalent. For example, these two requirements are equivalent according to the similarity metric:
R11a- the A320 neo optimal cruise altitude is FL380 in normal operating conditions versus
R11b- the A320 neo optimal cruise altitude is between FL 360 and 380, depending on weather conditions.
R11b is just more precise.

(2) **Use of more generic terms:** it is also frequent that two requirements differ only in a general purpose term or a business term where one is more generic than the other, or with a language level that is different. Detecting this situation requires an accurate domain ontology, not yet implemented in our system, and a real technical challenge in general.

(3) **Two or more differences between two requirements:** when two requirements have more than two groups of terms which are different, it turns out that in most cases they deal with different aspects of a given topic. The dissimilarity analysis should be revised so that the case of several observed differences is constrained, for example manner or temporal adverbs and different values or arithmetical expressions as advocated in 2.2(b) can co-exist, but not two groups of different values.

(4) **Presence of negative terms:** the negation, although not recommended in technical writing, or negatively oriented terms (verbs, adjectives) may appear in one requirement and not in the other, but these requirements are in fact similar to a large extent. For example:
R12a- Acid A must not be thrown in any standard garbage versus
R12b- Acid A must be thrown in a dedicated garbage.

(5) **Influence of the co-text:** requirements dealing with a given activity are often grouped under a title, subtitle, an enumeration or in a chart. Two arguments that belong to two different groups may seem to be incoherent, but if the context, given by the title or by the enumeration introduction head is different, then these two requirements may deal with different cases as in (1) and may not be incoherent. Co-text aspects are crucial in incoherence, but quite difficult to take into account because the link between a co-text and the requirement needs to be analyzed at the text level.

In terms of silence, two requirements have not been detected as incoherent because of:

(6) **implicit elements:** this is the case in the pair R10 (detected in a different way) where the implicit 'for the storage capacity' expression is unexpressed in pair a. The same situation is observed with pronouns, although their use should be very limited in technical authoring. Missing information prevents the similarity metric from mining requirements dealing with the same precise topic.

(7) **an external context:** Similarly to (5) above, but in the other direction, two requirements may be exactly identical but incoherent if the sections in which they appear have titles which are opposed in some ways. The incoherence may then be 'external' to the requirements. However the case structure evoked in (1) may also be considered here.

7. Conclusion and perspectives

We have presented in this paper the construction of a corpus of incoherent requirements and a preliminary categorization. We have restricted ourselves to incoherences which can be detected on a linguistic basis. Mining incoherences is really challenging as shown by the analysis provided for each category. A model and a simple implementation in TextCoop has been developed which provides relatively good results\enleadertwodots

This contribution opens new perspectives (1) on new forms of incoherence in texts and (2) on mining incoherent requirements, and arguments more generally. Our corpus examples show that incoherence may take a large diversity of forms.

A major difficulty is evaluation. Since it is not possible to have a test corpus where incoherent requirements have been identified manually, it is only possible to evaluate noise, but not silence. Both dimensions are important to evaluate the accuracy and also the linguistic adequacy and soundness of templates. In addition, requirement authors certainly do not want to be bothered by alerts about pairs of requirements which are not incoherent, they also wish to have an estimate of what the system found.

Acknowledgments

We thanks two anonymous reviewers for their useful comments that helped improved this article. We also thanks the companies who gave us access to their requirement documents in spite of their high confidentiality.

Appendix A. Supplementary data

Supplementary data related to this article can be found at http://dx.doi.org/10.1016/j.datak.2018.05.006.

References

[1] K. Ament, Single Sourcing. Building Modular Documentation, W. Andrew Pub, 2002.
[2] K. Asooja, G. Bordea, Using semantic frames for automatic annotation of regulatory texts, NLDB'16, Springer Lecture Notes, Salford, UK, 2016.
[3] P. Béguin, Design as a Mutual Learning Process between Users and Designers Interacting with Computers vol. 15, (2003) (6).
[4] A. Cruse, Lexical Semantics, Cambridge university Press, 1986.
[5] L. Fontan, P. Saint-Dizier, Analyzing the explanation structure of procedural texts: dealing with Advices and Warnings, STEP Conference, Venice, 2008.
[6] N.E. Fuchs, First-order reasoning for Attempto controlled English, Proceedings of the Second International Workshop on Controlled Natural Language, Springer, 2012.
[8] A. Freitas, J. Pereira, On the semantic representation and extraction of complex category descriptors, NLDB'14, Springer Lecture Notes, Montpellier, 2014.[9] M. Garnier, P. Saint-Dizier, Improving the use of English in requirements, J. IREB 6 (3) (2016).
[10] J.O. Grady, System Requirements Analysis, Academic Press, USA, 2006.
[11] E. Hull, K. Jackson, J. Dick, Requirements Engineering, Springer, 2011.
[12] J. Kang, P. Saint-Dizier, Discourse structure analysis for requirement mining, Int. J. Knowl. Content Develop. Technol. 3 (2) (2013).
[13] J. Kloetzer, S. De Saeger, Two-stage method for large-scale acquisition of contradiction pattern pairs using entailment, Proc EMNLP'13, 2013.
[14] T. Kuhn, A principled approach to grammars for controlled natural languages and predictive editors, J. Logic Lang. Inf. 22 (1) (2013).
[15] T. Kuhn, A survey and classification of controlled natural languages, Comput. Ling. 40 (1) (2014).
[18] M.C. De Marneffe, A.N. Rafferty, C.D. Manning, Finding contradictions in text, ACL-HLT'08, 2008.
[20] P. Saint-Dizier, Processing natural language arguments with the < TextCoop > platform, J. Argumentation Comput. 3 (1) (2012).
[21] P. Saint-Dizier, Challenges of Discourse Processing: the Case of Technical Documents, Cambridge Scholars, 2014.
[23] K.A. Schriver, Evaluating text quality: the continuum from text-focused to reader-focused methods, IEEE Trans. Prof. Commun. 32 (1989) 238–255.
[24] M. Unwalla, AECMA Simplified English, (2004) http://www.techscribe.co.uk/ta/aecma-simplified-english.pdf.
[25] E.H. Weiss, How to Write Usable User Documentation, The Oryx Press, Westport, 1991.
[26] R.H. Wojcik, E.H. James, Controlled Languages in Industry, in Survey of the State of the Art in Human Language Technology (Chapter 7), (1996) http://www.lt-world.org/hlt-survey/ltw-chapter7-6.pdf.
[27] A. Wyner, K. Angelov, C. Barzdins, D. Damljanovic, B. Davis, N. Fuchs, S. Hover, K. Jones, K. Kaljurand, T. Kuhn, M. Luts, J. Pool, M. Rosner, R. Schwitter, J. Sowa, On Controlled Natural Languages: Properties and Prospects, (2010) http://wyner.info/research/Papers/CNLP&P.pdf.