Risk Analysis and Prevention: LELIE, a Tool dedicated to Procedure and Requirement Authoring

Flore Barcellini (1), Camille Albert (2), Corinne Grosse (1), Patrick Saint-Dizier (2)

(1) CNAM-CRTD, 41 Rue Gay Lussac, Paris, France
(2) IRIT-CNRS, 118 route de Narbonne 31062 Toulouse cedex France,
Flore.Barcellini@cnam.fr, stdizier@irit.fr

Abstract

In this paper, we present a tool that detects business errors in technical documents such as procedures or requirements. The objective is to improve readability and to check for some elements of contents so that risks that could be entailed by misunderstandings or typos can be prevented. Based on a cognitive ergonomics analysis, we survey a number of frequently encountered types of errors and show how they can be detected using the <TextCoop> discourse analysis platform. We show how errors can be annotated, give figures on error frequencies and analyze how technical writers perceive our system.

Keywords: authoring tool, requirements, logic programming

1. Objectives

The main goal of the LELIE project is to produce an analysis and a piece of software based on language processing and artificial intelligence that detects and analyses potential risks of different kinds (first health and ecological, but also social and economical) in technical documents. We concentrate on procedural documents and on requirements (Hull et al. 2011) which are, by large, the main types of technical documents used in companies.

Given a set of procedures (e.g. production launch, maintenance) over a certain domain produced by a company, and possibly given some domain knowledge (ontology, terminology, lexical), the goal is to process these procedures and to annotate them wherever potential risks are identified. Procedure authors are then invited to revise these documents. Similarly, requirements, in particular those related to safety, often exhibit complex structures (e.g. public regulations, to cite the worse case): several embedded conditions, negation, pronouns, etc., which make their use difficult, especially in emergency situations. Indeed, procedures as well as safety requirements are dedicated to action: little space should be left to personal interpretations.

Risk analysis and prevention in LELIE is based on three levels of analysis, each of them potentially leading to errors made by operators in action:

- Detection of inappropriate ways of writing: complex expressions, implicit elements, complex references, scoping difficulties (connectors, conditionals), inappropriate granularity level, involving lexical, semantic and pragmatic levels, inappropriate domain style,
- Detection of domain incoherencies in procedures : detection of unusual ways of realizing an action (e.g. unusual instrument, equipment, product, unusual value such as temperature, length of treatment, etc.) w.r.t. similar actions in other procedures or to data extracted from technical documents,
- Confrontation of domain safety requirements with procedures to check if the required safety constraints are met.

The LELIE project is obviously very vast. We concentrate in this paper on the first task: the detection of inappropriate ways of authoring procedures and requirements. These two textual 'genres' share some characteristics since they both obey forms of operational style, but they also have their own specificities. Requirements are of two types: prevention requirements (how to avoid risks) and emergency ones (what to do when a problem occurs). We concentrate our investigations on industrial documents, while enriching our studies with a few large-public documents which are in general much more free and complex linguistically speaking. The documents we consider in this project range from short emergency notices to large procedures that may be more than 100 pages long. These large documents often follow authoring guidelines proper to the company which must also be taken into account and checked, similarly to grammatical and general style constraints.

Most industrial areas have now defined authoring recommendations on the way to elaborate, structure and write procedures of various kinds. However, our experience with technical writers shows that those recommendations are not very strictly followed in most situations. Our objective is to develop a tool that checks ill-formed structures w.r.t. these recommendations and general style considerations in procedures and requirements when they are written.

In addition, authoring guidelines do not specify all the aspects of document authoring: our investigations on author practices have indeed identified a number of recurrent errors which are linguistic or conceptual which are usually not specified in authoring guidelines. These errors are basically identified from the comprehension difficulties encountered by technicians in operation using these documents to realize a task or from technical writers themselves which are aware of the errors they should avoid.

We concentrate on those errors which are related to technical document authoring which are obviously not detected by standard text editors such as Microsoft Word or by profesional authoring tools.

2. The Situation and our contribution

Risk management and prevention is now a major issue. it is developed at several levels, in particular via probabilistic analysis of risks in complex situations (e.g. oil storage in natural caves). Detecting potential risks by analyzing business errors on written documents is a relatively new approach.
Authoring tools, in particular for simplified languages, have emerged two decades ago (e.g. one of the first investigated at Boeing, then at IBM: acrocheck/link, SPSS, see a synthesis at:
http://www.shu.ac.uk/prospectus/course/141/).
International style standards have been formulated, e.g. Hunt overrides, http://www.techknowledgecorp.com/help/tools.html), Madpak
(http://www.madcapsoftware.com/products/madpak/overview.aspx). (Ament 2002) and (Weiss 2000) developed a number of useful methodological elements for authoring technical documents and error identification and correction.
The originality of our approach is that :

- authoring recommendations are made flexible and depend on context, for example if negation is not allowed in instructions in general, there are however cases where it cannot be avoided because the positive counterpart cannot so easily be formulated, e.g. *do not throw the acid into the sewer*: but *throw where?*. Similarly, references are allowed if the referent is close and non-ambiguous. However, this requires some knowledge.

- following observations in ergonomics in the project, a specific effort is realized concerning the well-formedness (following grammatical and cognitive standards) of discourse structures and their regularity over entire documents (e.g. instruction or enumerations all written in the same way),

- the production of procedures includes some controls on contents, in particular action verb arguments, as indicated in the second objective above, via the Arias prototypical action base, e.g. avoiding typos or confusions among well syntactically and semantically identified entities such as instruments, products, equipments, values, etc.

- there is no real requirement analysis system based on language, that can check authoring recommendations. The main products are Doors and Doors-Trec, Objecteering, Reqtify and Craddle which are essentially text databases with query facilities and some traceability functions carried out via attributes. However, there are ongoing attempts to improve Doors possibilities with some forms of conceptual analysis based on predefined attributes, but this results in a relatively limited expressive power,

- the authoring tool includes facilities for French speaking authors who need to write in English, supporting typical errors they make via 'language transfer' (Garnier 2010, 2011). This point is not addressed here.

This project, LELIE, is based on the TextCoop system (Saint-Dizier, 2012), a system dedicated to language analysis, in particular discourse (including the taking into account of long-distance dependencies). This project also includes the Arias action knowledge base that stores prototypical actions in context, and can update them. It also includes an ASP solver to check for various forms of incoherence and incompleteness. The kernel of the system is written in Prolog SWI, with interfaces in Java. The project is at the moment realized for French, an English version is under development.
The system is organized according to the following principles:

- the system is parameterized: the technical writer may choose the error types he wants to be checked, and the severity level for each error type when there are several such levels (for example there are several levels of severity associated with fuzzy terms which indeed show several levels of fuzziness),

- the system simply tags elements identified as errors, the correction is left to the author. However, some help or guidelines are offered. For example, guidelines to reformulate a negative sentence into a positive one are proposed,

- the way errors are displayed can be customized to the writer's habits.

We present below a kernel system that deals with the most frequent and common errors made by technical writers independently of the technical domain. This kernel needs an in-depth customization to the domain at stake. For example, the verbs used or the terminological preferences must be implemented for each industrial context. Our system offers the control operations, but these need to be associated with domain data.
Finally, to avoid the variability of document formats, the system input is an abstract document with a minimal number of XML tags as required by the error detection rules. Managing and transforming the original text formats into this abstract format is not dealth with here.

3. Categorizing language and conceptual errors found in technical documents

In spite of several levels of human proofreading and validation, it turns out that texts still contain a large number of situations where recommendations are not followed. Reasons are analyzed in e.g. (Beguin, 2003), (Mollo et al. 2004, 2008).
Via ergonomics analysis of the activity of technical writers, we have identified several layers of recurrent error types, which are not in general treated by standard text editors such as Word or Visio, the favorite editors for procedures.
Here is a list of categories of errors we have identified. Some errors are relevant for a whole document, whereas others must only be detected in precise constructions (e.g. in instructions, which are the most constrained constructions):

- general layout of the document: size of sentences, paragraphs, and of the various forms of enumerations, homogeneity of typography, structure of titles, presence of expected structures such as summary, but also text global organization following style recommendations (expressed in TextCoop via a grammar), etc.
- morphology: in general passive constructions and future tenses must be avoided in instructions,
- lexical aspects: fuzzy terms, inappropriate terms such as deverbals, light verb constructions or modals in instructions, detection of terms which cannot be associated, in particular via conjunctions. This requires typing lexical data.
- grammatical complexity: the system checks for various forms of negation, referential forms, sequences of conditional expressions, long sequences of coordination, complex noun complements, and relative clause embeddings. All these constructions often make documents difficult to understand,
- uniformity of style over a set of instructions, over titles and various lists of equipments, uniformity of expression of safety warnings and advice,
- correct position in the document of specific fields: safety precautions, prerequisites, etc.
- structure completeness, in particular completeness of case enumerations w.r.t. to known data, completeness of equipment enumerations, via the Arias action base,
- regular form of requirements: context of application properly written (e.g. via conditions) followed by a set of instructions,
- incorrect domain value, as detected by Arias.

When a text is analyzed, the system annotates the original document (which is in our current implementation a plain text, a Word or an XML document): revisions are only made by technical writers. When documents are inspected in batch mode, a score of the document global quality level may be produced.
Tags are customized to authors, they are made explicit and easy to understand, as shown in the following two short examples where a possible way of tagging is proposed:

Move the vacuum compressor pump <to the area: WHICH ONE?>. <For instance: EXAMPLE of WHAT?>, close to the water storage or to the pool. <It: WHO?> should be on a higher level than <the source of water : VAGUE?> and <probably: VAGUE?> 3 feet away. If you are working on a pool, <it: WHO?> connects to the skimmer line.

The vagueness of the 'example' is due to the previous indeterminacy of the 'area': TextCoop rules allows, via the Dislog language to deal with non-adjacent structures in texts. The second example, translated from French, shows the detection of enumerations which are not homogeneous:

- Open the canal JD34.
- <you must maintain: STYLE NOT uniform> pressure during 1 minute <at least: VAGUE?>.
- then close <the flood-gate: WHICH ONE?> quickly at 100 %.
- <Do not: NEGATION> open <it WHAT?> again in the next minute.

Besides tags which must be as explicit as possible, colors indicate the severity level for the error considered (the same error, e.g. use of fuzzy term, can have several severity levels). The most severe errors must be corrected first. At the moment, we propose four levels of severity:

1. ERROR: must be corrected,
2. AVOID: preferably avoid this usage, think about an alternative,
3. CHECK: this is not really bad, but it is recommended to make sure this is clear. This level is also used to ask users make sure that argument values are correct, when non-standard ones are found,
4. ADVICE: possibly not the best language realization, but this is probably a minor problem. It is not clear that there are alternatives. However, if there are, reformulations are welcome.

We evaluate in section 5.5 how text authors react to these recommendations: how many errors are indeed corrected, how severe they are found, and which errors are left unchanged. An illustration is given in Figure 1 at the end of this paper.

4. Linguistic Model and Implementation

Let us now present our correction system and its performances. It is based on a system of usage rules that detects ill-formed constructions, rules are written by hand from ergonomics observations and recommendations. Rules are developed on the <TextCoop> platform (Saint-Dizier 2012) and written in the Dislog language. <TextCoop> is based on logic programming, structures and errors are detected on the basis of rewrite rules. These rules may be associated with various forms of reasoning and knowledge (Kintsch 1988) (Grosz et al. 1986).
The kernel system includes about 75% of the set of errors observed in writing and reading phases by the ergonomists of the project. This is relatively large. However, some final tuning will remain to be done for each company when the system is deployed, in order to take into account company usages, in particular lexical. The errors not included are essentially related to the style imposed by companies, which will be developed separately, and to the domain terminology preferences and practices, which can be very diverse. Portability and customization are obviously central issues in this work and a methodology must be realized to evaluate techniques, feasibility and costs.
For each error type, a cluster of detection rules is produced. The system first makes a discourse analysis of the document following the RST theory and principles (Man et al. 1988) (Marcu 1997, 2000). Our rules in Dislog identify in

particular: titles, pre-requisites, warnings, advice, instructions, goals, purposes, conditions, etc. in fact most discourse structures related to explanation as found in this type of document (Bourse and Saint-Dizier 2012). This is realized also by a set of rule clusters, executed sequentially, in a precise order implemented by means of a cascade of rules. Then, the various errors are checked one after the other, using another cascade. The type of discourse structure to be inspected is specified in the rules using the tags which have been produced during the discourse analysis.
Lexical resources have been developed accordingly. These contain, for example:

- standard terms such as determiners, negation, pronouns, conjunctions, modals, etc.
- lists of fuzzy terms, with a severity level,
- lists of application dependent verbs and deverbals, together with their subcategorization frames, selectional restrictions, and argument types, this is associated with the Arias knowledge base contents,
- lists of positively or negatively oriented terms respectively used to identify advice and warnings.

In addition, rules are associated with a morphological analyzer, a POS tagger and a number 'local' grammars, included in the <TextCoop> environment.

5. Error synthesis and analysis

Let us now report here the errors which have received an in-depth treatment in the system kernel. We focus on lexical, grammatical and stylistic errors and give their main characteristics.
We indicate the following main features: the type of error, its severity (1 to 4, 4 being the highest), its global frequency (1 to 4) and its portability (1 to 4, 4 meaning easy to adapt to any context).

5.1. Lexical-based errors

The following criteria have been evaluated:

error type	severity	frequency	portability
fuzzy term	3	3	3
verb arguments	3	2	1
preferred terms	2	2	1
domain terms terms to avoid	3	2	1
deverbals	2	3	2
modals	2	2	1
light verb constructions	2	1	2
verb diversity	3	3	3
pronouns	4	4	1

'Verb argument' relates controls on the nature and completeness of verb arguments realizations for recurrent instructions. In general objects, instruments and values (Ph, weights, durations, volts, etc.) are checked. This is realized via a simple verb argument recognition grammar included in <TextCoop> environment. Semantic controls require the use of a domain terminology or ontology. They are in general only partial.
'Preferred terms' means that the term found must be reconsidered and that there is a preferred term, often a domain term. This is identified via specific relations in the domain terminology.
'Terms to avoid' are terms judged inappropriate in the domain (e.g. instead of *control* use *make sure that*). These are often very specific and need to be specified in the system for each company, activity or even group of technical writer.
In general, modals, light verb constructions and deverbals must be avoided unless they are really part of the domain language. Modals may appear in warnings and advice, not in instructions where they somewhat hide the injunctive character of an instruction. Light verbs and deverbals introduce some linguistic complexity which is often judged unnecessary.
'Verb diversity' is a measure of the number of distinct verbs used in instructions. It is often recommended to limit the set of verbs to a minimum which can be as low as 20 or 30.
'Pronouns' indicates the use of a pronouns where the antecedent may not be so easy to identify. To limit complexity of the treatment, the system tags pronouns which appear at the beginning of sentences, these seem to be the most difficult to relate to an antecedent.

5.2. Grammar-based errors

The following criteria have been evaluated:

error type	severity	frequency	portability
negation	3	4	1
embedded clauses	4	3	1
term position	3	3	1
coordination	4	3	1
noun complements	3	3	4
passives	2	3	1
future	2	2	1

'Negation' includes double as well as simple negation, the latter with a much lower severity level. Negative forms can be very diverse. In instructions, standard forms such as *not, do not, never* are tagged, negative terms such as *avoid* are also tagged since they convey a negative meaning. Transforming a negative expression into a positive expression is often very challenging as illustrated above. For that purpose, we propose a few correction schemas. However, the best practice is to ask the technical writer to make sure that the negative expression is clear (or to make it as clear as possible) if he does not want to change it.
'Embedded clauses refers to relative clauses which are embedded in instructions in particular. Their understanding may lead to misinterpretations.
'Term position' refers to preferences which must be detected, in particular a number of company styles require verbs to appear first in instructions (possibly with negation). In other situations, conditions must appear first. Initial position is basically checked, we do not at the moment control terms or constructions which are in final positions in instructions (e.g. as low level goals are in general, to clearly indicate they are local).

'Coordination' and 'noun complements' indicated a too high level of these constructions in any part of the document.

As can be noted, in contrast with lexical errors, grammar-based errors are generic and require low portability costs.

5.3. Style and Discourse errors

This level includes a number of criteria that depend on the style of the company. We simply survez three generic ones here.

error type	severity	frequency	portability
sentence length	3	3	1
references	4	3	1
enumerations	3	4	3

Documents often contain very long sentences. We propose here several severity levels depending on the length, which can be parameterized. 'References' indicates references to other document portions which are not adjacent, they therefore require a document traversal which may not be comfortable for technicians in operation.

Finally, 'enumerations' evaluates the regularity of the expressions in a list of enumerated items. Since this task is very complex and requires flexibility, a number of criteria are checked and a diagnosis is produced. For example, the contents of the three first terms is checked and a regularity measure is produced that depends on the terms and their position. In order to avoid complex parsing or access to a large variety of lexical data, basically closed word classes and verbs are considered. These are often the most frequent categories found starting enumerations. Nouns could also be checked, however, via the domain terminology.

5.4. Errors as found in texts

To evaluate our error detection system we first evaluate its impact in texts. We considered about 300 of full text pages (procedures and requirements) from three main French companies, involved in three very different industrial areas (energy, chemistry and transportation). For confidentiality reasons, let us call them A, B, and C. These companies have quite a large group of technical writers (about 30 to 50 each, with well-trained staff and beginners). Each company has very different authoring constraints and validation processes. Documents are in French for the moment, the system is under development for English.

The results given below show the amount of errors that have been detected. The figures for each error is an average for 1000 lines of text. As it can be noted, the error rate is far from being negligible: about one error every two lines, not counting domain related errors. This clearly shows the importance of our system. Only errors which do not require a high portability cost are reported in this chart since we use the kernel system.

error type	global result	A	B	C
fuzzy term	66	44	89	49
deverbals	29	24	14	42
modals	5	0	12	0
light verbs	2	2	2	3
verb diversity	average	low	high	low
pronouns	22	4	48	2
negation	52	8	109	9
embedded clauses	7	0	5	0
term position	56	57	82	33
coordination	6	0	10	0
noun complements	36	28	62	15
passives	34	16	72	4
future	2	2	4	1
sentence length	108	16	221	24
enumerations	average	low	high	average
references	13	33	22	2

It is difficult at this stage to analyze the reasons for high levels of errors in some cases. It must also be taken into account that the technicians that use (or are supposed to use) these documents probably have very different technical levels, some of them being capable of understanding complex statements. It must also be taken into account that some documents have an everyday use whereas others serve in emergency or infrequent situations. These latter certainly need a more accurate writing and proofreading.

5.5. User reaction

Given these errors, it is now of much interest to measure how technical writers react to the errors which are displayed in their texts. An objective measure (besides their satisfaction of dissatisfaction) is to count the number of errors which have been corrected. Several strategies can be deployed by technical writers at this level:

1. an error has been found relevant and has indeed been corrected appropriately. Futhermore, the writer may wish to keep track of his correction for further situations or for other writers, as an example,
2. an error has been found relevant, but considering the text segment at stake, the writer realizes that the correction must be done on a larger scale,
3. an error has been found relevant and has been corrected, but it has generated another error,
4. an error has been found relevant but the author (and his colleagues) has no correction in mind, or he does not fully understand the error. The error is left unchanged in the text possibly for later inspection. The writer may also look for similar errors in the text to see if and how they have been corrected,
5. an error that the writer does not want to correct, for various reasons, in particular because it is minor or it does not alter the understanding of the sentence,
6. an error indicated by the system, but that turns out to be a wrong diagnosis: the text span is perfectly correct.

Point 4 relates a frequent situation. For that purpose, although we cannot anticipate corrections for each situations, we can nevertheless propose partial correction schemas. We review below, in French and English (glosses) a few such schemas.

- Negation: (1) s'assurer que X ne V (where X is a noun or an NP and V a verb: ensure that X does not V) rewrites into: éviter que X V (avoid X to V). (2) X ne V que lorsque rewrites into: X V seulement lorsque (not appropriate in English): *Cette vanne n'est utilisée que lorsque..., cette vanne s'utilise seulement lorsque.* However, in this latter example, a middle reflexive construction is introduced.

- Deverbals, light verb constructions and passives: the direct verb form can be proposed, together with a reorganization of the verb arguments.

- Complex groups of conditions or contexts in a sentence: reformulation as the action to realize followed by a list (with an appropriate typography) of the conditions *in case of high pressure, or if the temperature is higher that 50 degrees, stop the injection, or if the bore concentration goes above 5%.* can be rewritten into:
  stop the injection:
  (1) in case of high pressure,
  (2) in case the temperature is higher that 50 degrees,
  or (3) in case the bore concentration is above 5%.

The first experimental results we get concerning user reaction are the following, given a few correction schemas. Numbers refer to the six above cases. Results are based on small samples of corrected texts submitted to writers and are still quite exploratory. We noted a great interest from technical writers and their desire to improve their texts as much as possible, sometimes beyond the errors which have been detected.

writer's attitude	1	2	3	4	5	6
situation frequency (%)	31	26	8	15	8	12

6. Conclusion and perspectives

We have presented in this paper the first phase of the LELIE project: detecting authoring errors in technical documents that may lead to risks. We surveyed here a number of errors: lexical, business, grammatical, and stylistic. Errors have been identified from ergonomics investigations. The system is now fully implemented on the <TextCoop> platform and has been evaluated on a number of documents. It is then of much interest to evaluate user's reactions.
We have implemented the system kernel. The main challenge ahead of us is the customization to a given industrial context. This includes, among others:

- accurately testing the system on the company's documents so as to filter out a few remaining odd error detections,

- introducing the domain knowledge via the domain ontology and terminology, and enhancing the rules we have developed to take every aspect into account,

- analyzing and incorporating into the system the authoring guidelines proper to the company that may have an impact on understanding and therefore on the emergence of risks,

- implementing the interfaces between the writer's documents and our system, with the abstract intermediate representation we have defined,

- customizing the tags expressing errors to the users profiles and expectations, and enhancing correction schemas.

When sufficiently operational, the kernel of the system will be made available on line, and probably the code will be available in opensource mode or via a free or low cost license.

7. Acknowledgements

This project is funded by the French national Research agency ANR, under the Emergence programme. We also thanks reviewers for their comments and the companies that showed a strong interest in our project, let us access to their technical documents and allowed us to observed their technical writers. For confidentiality reasons, these companies are not mentioned explicitly here.

8. References

Ament, K., Single Sourcing. Building modular documentation, W. Andrew Pub, 2002.

Béguin, P. Design as a mutual learning process between users and designers Interacting with computers, 15 (6), 2003.

Bourse, S., Saint-Dizier, P., A Repository of Rules and Lexical Resources for Discourse Structure Analysis: the Case of Explanation Structures, LREC 2012, Istanbul.

Garnier, M., Correcting errors produced by French speakers: the case of misplaced adverbs, Calico, Amherst, 2010.

Garnier, M., Correcting errors in N+N structures in the production of French users of English, EuroCall, Nottingham, 2011.

Grosz, B., Sidner, C., 1986. Attention, intention and the structure of discourse, Computational Linguistics 12(3).

Kintsch, W., 1988. *The Role of Knowledge in Discourse Comprehension: A Construction-Integration Model*, Psychological Review, vol 95-2.

Hull, E., Jackson, K., Dick, J., Requirements Enginnering, Springer, 2011.

Mann, W., Thompson, S., 1988. Rhetorical Structure Theory: Towards a Functional Theory of Text Organisation, *TEXT* 8 (3) pp. 243-281. Thompson, S.A. (eds), 1992. Discourse Description: diverse linguistic analyses of a fund raising text, John Benjamins.

Marcu, D., 1997. The Rhetorical Parsing of Natural Language Texts, ACL'97.

Marcu, D., 2000. *The Theory and Practice of Discourse Parsing and Summarization*, MIT Press.

Mollo, V., Falzon, P., Auto and allo-confrontation as tools for reflective activities. Applied Ergonomics, 35 (6), 531-540, 2004.

Mollo, V., Falzon, P. , The development of collective reliability: a study of therapeutic decision-making. Theoretical Issues in Ergonomics Science, 9(3), 223-254, 2008.

Saint-Dizier, P., 2012. Processing Natural Language Arguments with the <TextCoop> Platform, *Journal of Argumentation and Computation*.

Weiss, E.H., Writing remedies. Practical exercises for technical writing, Oryx Press, 2000.

CONSIGNES D'UTILISATION DES SORBONNES DE 2R1

Les sorbonnes disposent à présent d'un système à deux vitesses de fonctionnement grâce à la pose d'un variateur de fréquences couplé avec une platine de commande (ou contrôleur de débit) et ***d'****un contact de position* ***à*** *la fermeture* ***de*** CONSEIL: trop de compléments de noms *la glace relevable.* EVITER: phrase trop longue .

*Lorsque la sorbonne fonctionne avec l'écran en position basse, le débit d'air n'*CONSEIL: négation utile ? *est que de 300m*^{3}*/heure* ***environ*** ERREUR: terme flou quantité. *Dès que l'opérateur lève l'écran, un contact de position se libère et le variateur électronique de fréquences modifie la fréquence de fonctionnement pour atteindre 900 à 1000m*^{3}*/heure.* EVITER: phrase trop longue

- ✓ Lors d'une expérience qui **ne** CONSEIL: négation utile ? nécessite pas de manipulation, il est obligatoire de baisser l'écran **au maximum** ERREUR: terme flou manière. Aucun élément sur la paillasse ne CONSEIL: négation utile ? doit empêcher l'écran d'atteindre sa position basse maximale.

- ✓ Lorsque la sorbonne est à l'arrêt, un voyant rouge clignote pour le signaler. Il VERIFIER: référence claire pronom clignotera également en cas de panne de l'extracteur ou tout autre incident.

- ✓ Lors de la mise en route de l'extracteur, passé **quelques** secondes ERREUR: terme flou quantité pour la mise en dépression de l'enceinte, les leds vertes apparaissent. EVITER: phrase trop longue La hauteur de travail de la face avant est de 40cm par rapport au plan de la paillasse.

- ✓ Lorsque la manipulation sous sorbonne est terminée, il convient de ne pas arrêter l'extracteur immédiatement mais au contraire d'attendre quelques minutes de façon à ce que tous les polluants présent dans le tube soient évacués. EVITER: phrase trop longue Pendant cette période, l'écran devra EVITER: futur être baissé EVITER: passif au **maximum**. ERREUR: terme flou manière

- ✓ Lors de l'ouverture de l'écran au dessus de 40 cm, une alarme visuelle par led rouge clignote pour signaler que les vitesses d'air en façade sont passées sous les 0.4m/s EVITER: phrase trop longue. Au bout d'une temporisation de 30 secondes **environ** ERREUR: terme flou quantité, l'alarme sonore se déclenche. Il suffit alors de ramener VERIFIER: verbe en début instruction l'écran à la hauteur de 40 cm pour arrêter l'alarme visuelle et sonore.

- ✓ <u>Exceptionnellement,</u> en cas de manipulation en hauteur dans la sorbonne, vous avez la possibilité de stopper VERIFIER: verbe en début instruction temporairement l'alarme sonore par un bouton sur la platine de commande.

- ✓ Les écrans **ne** CONSEIL: négation utile ? sont pas en verre. Il est absolument interdit d'y écrire dessus.
  CONSEIL: énumération : pas de diagnostic établi, peut être à revoir.

Rédacteur : SPS AA
Date de création : 06/07/10
Date de mise à jour : 17/03/2011

Figure 1: An output sample.