Abstract

This paper argues for looking at Controlled Languages (CL) from a Natural Language Generation (NLG) perspective. We show that CLs are used in a normative environment in which different textual modules can be identified, each having its own set of rules constraining the text. These rules can be used as a basis for natural language generation. These ideas were tested in a proof of concept generator for the domain of aircraft maintenance manuals.

1 What is a Controlled Language?

Controlled Languages (CLs) result from a growing concern about technical documentation quality and translation, both human and automatic. A CL consists of a glossary and writing rules for the linguistic aspect of the documentation. These rules are given as recommendations or prohibitions for both the lexicon and the grammar. Currently, most CLs are varieties of controlled English which derive from the Caterpillar Tractor Company Fundamenta English that was elaborated in the sixties (Sheurs and Adriaens, 1992). However, CLs are presently being developed for German, Swedish and French.

Technical writers find it difficult to comply with the writing rules of a CL which are often hard to justify (CLA, 1996). For them, a CL is seen as an additional constraint on an already complex task. This is why tools have been developed for CL users, the best known being conformity checkers/controllers such as AlethCL or SECC (CLA, 1996).

A writer expects that a checking tool should not only detect error but also propose a CL-conformable expression. Nasr (1996) who worked on the problem of CL reformulation underlined that this task of reformulation may be very difficult to correct: for example, sentences that are longer than the prescribed number of words. So there is little hope that human writers will ever produce documents complying strictly with a CL even with the help of a conformity checker. We argue that it may be more promising to use NLG technology to generate the documentation in a CL instead of analyzing it afterwards, as it is the case with a conformity checker. Few researchers have looked at CLs from a generation point of view (Nasr, 1996; Hartley and Paris, 1996), but we think that there are very compelling reasons for taking a generation perspective in addition to the advantages of NLG for CLs that will be presented in section 3.

† As CLs can be viewed as linguistic specifications for human beings, it seems natural to consider them as specifications for the linguistic component of an NLG system.
† CL writing specifications come on top of other writing norms which deal with document structuring, for example, in the aeronautical industry. CLs such as Simplified English (SE) (AEC, 1995) and Francais Rationalise (FR) (GIRAS, 1996) extend the ATA 100 norms (Bur, 1995) which describe the division of the document into chapters, sections, subsections, reflecting a tree-structured functional organization of the airplane: a chapter corresponds to a system (e.g., main rotor), a section to a sub-system (e.g., gearbox), a subsection to a sub-sub-system (set of gears), and so on. Over this thematic structure is added a communicative structure to fulfill two main goals: describe all systems of the airplane and prescribe all maintenance instructions for the airplane. The norms of the ATA can be viewed as specifications for the text structuring component of an NLG system.

† The thematic and communicative structuring of the document must also conform to a systematic non-linear page-numbering system and strict formatting rules using SGML tags. These constraints can be viewed as specifications for the layout component of an NLG system.

So we claim that CLs should not be considered outside the context of the production of complex structured documents which naturally raise the question of the automatic generation of this documentation given some formal representation. This is claimed by Lux (Lux, 1998) to reorient the notion of a CL. Her study has shown that only a few syntactic constraints (e.g., coordination constraints) are applicable to the whole document. Most constraints are only valid for sub-parts of the document, identified as textual modules. Each textual module has a particular communicative goal and a precise thematic according to the ATA 100 norms. It can be divided into smaller modules. For example, the Task module is divided into simple Sub-Task modules which are themselves composed of simple Instruction modules. From a linguistic point of view, a textual module uses only a controlled sub-language. Lux thus extended FR to a new CL called FREM (Francais Rationalise Etendue Mo dulaire) comprising many CLs, each having its own syntactic rules for a specific textual module. She also performed a corpus study showing that the same textual modules could be identified for both French and English. It should thus be possible to re-module similarly what has been done to FR with FREM. In this paper, we therefore introduce the notion of an Extended Mo dular Contro lled Language (EMCL) which first defines some general rules and then some more specific ones for each textual module. We now look at the problem of automatically generating technical documentation complying with both structure and organization such as ATA 100 and to the rules of an EMCL.

2 How to generate technical documentation?

We assume that a generation system can be divided into a What to say and How to say it component, even though this may be considered a gross simplification.

2.1 What to say component

The main difficulty for NLG in a real environment lies in knowledge modeling. For aircraft maintenance manuals, existing ontologies could probably be reused, but even then, the modeling efforts required are huge. Nevertheless, assumptions about possible design forms which are sequentially presented to the user (as in Drafte (Paris et al., 1995)) through which the technical writer provides the information to convey in an appropriate formalism. These forms can be derived directly from the tree-structured document given in the ATA norms. The goal is that the technical writer can fill in these forms, which describe the technical documentation already properly structured in an abstract language instead of a natural one. In a general text generation setting, simple forms are used to elicit information from users to describe what is to be said, and this seems like a difficult task, but in the context of technical writing, information is almost already prescribed, and forms are thus a simple way of complying with the rules of a CL. Indeed, in a number of common environments, forms are frequently used to elicit information from users. This input can then be processed by the How to say it and layout components.
The writers who find it very difficult to comply with the rules of a CL have no problem complying with the ATA 100 norms, thereby producing documents with the right thematic and communicative structuration and sometimes making an illustration of observations made in psycholinguistics. Levitt (Levitt, 1989, p. 9) describes a model of the speaker’s activity in which choices in the What to say component are conscious, while choices in the How to say it component are automatic. This model helps understand some of the difficulties that CL users face. A CL forces the writer to become conscious of behavioral mechanisms that are usually automatic. The writer is thus distracted from choices made earlier in her/his writing task. So s/he often ends up writing in the way it has to be written but not exactly what had to be written, thus defeating the whole purpose of a CL which was meant to produce a better expression of the information. This model also explains why a human writer has less difficulties following the ATA norms: this part of the job is replaced by filling in some information in the forms that are presented.

To sum up, the What to say component requires a modelization of the domain model and the design of a series of forms to be filled. A human writer using the NLG scenario has to be conscious of all the necessary information for generation is provided.

2.2 How to say it component

In this section, it is assumed that if a CL is infact an EMCL such as FREM, a specific way to say it component is designed for each textual module but always retains the same formalism.

The lexicosed in the How to say it component should be exactly the one enforced by the CL. Similarly, the syntactic constructions and the discourse structure in this component should correspond to the set of allowed constructions in the CL. This can simplify some lexical and even discourse choices to be made with the generator system and thus ensure that the generated text complies with the rules of the CL. However, many writing rules in a CL place particular syntactic restrictions on the use of a given lexical item, for example, in FREM, the use of certain verb phrases is not allowed. This makes it easy to follow the evolution of rules in an (EM)CL. For example, if the rule to write an Instruction
Changes from "Put a verb in the infinitive" to "Insert an imperative", then this must be changed everywhere in the lexicalized grammar. Using the metagrammar we can achieve this quite easily because of the hierarchical organization of a LTAG: with only one rule an imperative can be allowed and an infinitive is disallowed (in main clause) for every verb, whatever its argument structure and syntactic construction.

G-TAG thus seems a good candidate for producing technical documentation complying with the constraints of an (EM) CL. A technical documentation generator prototype in the aeronautical domain is described in Section 3. This generator is written in Flaubert, an implementation of G-TAG (Danlos and Meunier, 1996). The How to say component would have to be completed by adding a layout component complying with the norms of ATA 100. We should also provide revision tools to allow the writer to netun the final text.

So, automatic generation of technical documentation became technically possible. The technical writer is willing to use a form in which principles should be less demanding than learning rules of an (EM) CL. This approach also has the advantages described in the next section.

3 Advantages of automatic generation of technical documentation

3.1 Multilinguality

One of the major assets of NLG is its capacity to simultaneously generate text in several languages and to regenerate updates of the same text using a single input representation, thus ensuring coherence among the generated texts.

Until now, CLs have dealt with multilinguality by means of the translation hypothesis. It is therefore surprising that this was developed by adapting SE, in order to ease the translation from French to English. It authorizes the use of everything that can also be written in FR and can be translated into SE. From this point of view, the definition of a source CL depends on the definition of a target CL. Developments of CLs are more likely to select structures which can be easily retranslated into the target CL. What then happens if CLs and source CLs are structurally different? This can lead to a situation where CLs impose cumbersome writing style that could be different from the conventions shared by native speakers of FR, thereby contradicting the aim of enhancing understandability.

Rules of an (EM) CL should be elaborated without multilingual considerations. Their definition should primarily aim to enhance the characteristics of the language being used to write the documentation in the aeronautical domain must respect the functional decomposition of the aeronautical plane and that the same text can be used in many languages. This means that nothing has to be changed in the What to say component (Section 2.1) going from one language to the other. Only the How to say component (Section 2.2) needs to be adapted to the target (EM) CL which should be monolingually defined.

3.2 NLG as an aid for testing and developing CLs

An NLG system can provide concrete assistance for testing and for the development of a CL. An NLG system that integrates the CL constraints can help discover contradictions in the CL definition. As illustrated by an example in the following, it is possible to discover that a FR lexical rule is not conform to a FR lexical rule, according to which "empêcher l'oxygène" should be replaced by "empêcher l'oxygène" by an infinitive clause.

Empêcher l'oxygène de s'accumuler (Prevent the oxygen from accumulating) does not conform to a FR lexical rule, according to which "empêcher l'oxygène" should be replaced by "empêcher l'oxygène" by an infinitive clause.

Empêcher l'accumulation d'oxygène (Prevent oxygen accumulation) does not con-
Figure 1: Event graph given as input to Flaubert. In the prototype, this information is entered in textual form.

Form to FR lexicon according to which verbs ‘accumuler’ (accumulate) should be used instead of the noun ‘accumulation’. † Empêchez que l’oxygène ne s’accumule (Prevent that the oxygen accumulates) does not conform to the writing rule that forbids the use of the subjunctive mode.

So we come to a dead end. We want to use the verb ‘empêcher’ (prevent). This problem can be detected automatically by the NLG system and an appropriate fix be made in the grammar.

NLG can help strengthen a CL’s claim to lead to more homogeneous texts, which is equivalent to forbidding certain paraphrases. NLG precisely deals with paraphrases, for some inputs a NLG system will produce several texts. In this way, NLG helps identify which paraphrases are possible in the CL. In practice when an NLG system produces several texts for one input, it raises the question for the CL developer: Should a constraint be added to the CL definition in order to forbid one of these texts?  

4 Proof of concept generator

The previous sections have argued for the interest of dealing with CL from an NLG perspective which our knowledge had never been examined in such detail. To further pursue this, Lux (Lux, 1998) has developed a prototype of concept generators using Flaubert (Meunier, 1997; Meunier and Danlos, 1998) to see how these theoretical concerns could be applied in practice.
The generator can produce text about tensubtasks in FREM. Thesetask comprise fromtwo toeleven instructions illustrating differentsubtask types such as: simple instruction with a goal, simple instruction with a condition, complex instruction with simultaneous actions, etc. They involvethesetypinguticationsuch as infiniteseminalisation, etc.

Inputtotheprotoype are event graphs such as shownin Figure 2. The outputtext format inFrench as shown in Figure 2 which was generated from Figure 1. InLux’s prototype, the event graphswere hand coded, but now Flaubert has been rewritten in CLEF (Meunier, 1999; Meunier and Reyes, 1999) which has a bettergraphical interface.

The outputtext is a sub-task including different instructionstypes (only the firstthree instruction region in the Figures) to perform by the same person (e.g. U0). FREM defines what connections from each instruction type (e.g. conjunction and in structuralsimultaneous actions).

The generationof noun groups for the object(O), ingredients(Ing) and tools(T) rely on a mapping table between these labels and their denotations. It was a temporary solutions to problemoutside the scope of the prototype. We should have reliedon existing nomenclatures for tools and ingredients, and on the fact that objects are systematically presented in drawings associated with various subtasks. O5, called "segment d'arrêt", is labeled (5) on the drawing associated with the example above. In a graphical interfaceenvironment, authors would select these objects linked to a controlledterminology database.

This proof of concept generation worked well for purposed of testing our theoretical ideas, but unfortunately could not be evaluated in a realistic textproduction environment. Our sponsors were interested in these results; we have produced but changesin their organisation made it impossible to carry further investigations. We intend to further pursue research and use the new implementation of Flaubert to generate controlled languagenon the area of application while keeping the concept of an extendedmodular CL.

5 Conclusion

This paper has argued that linguistic norms imposed by CLs shouldnot be considered isolation. They are only a part of a set of more comprehensive norms on the document structure and layout. This insight leadsto define a notion of textural modules with their own linguistic norms, and to envisage the generation of technical documentation using an extended modularcontrolled language (EMCL). Norms for document structure in such as ATA100, its linguisticcharacteristics and its output requirements may be defined respectively defined in the text structuring, linguistic and the layout component of an NLG system.

We have also shown that generation of view can help define the definition of an EMCL. The EMCL can be defined monolingually and multilingually and obtained through NLG. These ideas were tested within a proof of concept text generation in the domain of aircraft maintenance manuals.

Acknowledgment

We thank our former colleagues at Aerospatiale Research Center and Frédéric Meunier who implemented Flaubert. We also thank Elliott Macklow who suggested many improvements to this paper.
References

AECMA Document PSC-85-1659A,F95. Simplified English Standard, a guide for the preparation of Aircraft Maintenance Documentation in the International Aerospace Maintenance Language.

Bureau de Normalisation de l’Aéronautique et de l’Espace (BNAE), Issy-les-Moulineaux, 1995. Specification ATA no 100, traduction française Specification for Manufacturer’s Technical Data — ATA Specification 100, October.

M.-H. Candito. 1996. A principle-based hierarchical representation of LTAGs. In Proceedings of the 16th International Conference on Computational Linguistics, pages 194-199, Copenhagen.

CLAW. 1996. Proceedings of the First International Workshop on Controlled Language Application (CLAW), Leuven.

L. Danlos and F. Meunier. 1996. G-TAG, un formalisme pour la génération de texte présentation d’applications industrielles. Actes du colloque informatique Langue Naturellement.

L. Danlos. 1998. G-TAG: un formalisme lexicalisé de génération de texte inspiré de l’arbre d’adjonction TAG. Thèse de doctorat en informatique, Université de Nantes.

M. Emorine. 1994. Projet de recherche sur la modélisation de la formule automatique des expressions verbales du français dans un rapport technique université de Clermont II. GIPAS. 1996. Guide du rédacteur partielle: Francisation d’un texte technique, GIPAS, Paris.

A. Hartley and C. Paris, 1996. Le texte procédural: langage, action et cognition. Thèse de doctorat en linguistique, université Paris.

F. Meunier and L. Danlos. 1998. FLAUBER T: un user-friendly system for multilingual text generation. In Proceedings of the First International Workshop on Natural Language Generation (INLG’98), pages 284-287, Niagara-on-the-Lake.

F. Meunier and R. Reyes. 1999. Plate-forme de développement de générateurs multilingues. In Actes de la conférence de Génération Automatique de Texte GAT’99, pages 145-155, Grenoble, France.

F. Meunier. 1997. Implémentation de G-TAG, formalism pour la génération inspirée des grammaires d’arbres adjointes. Thèse de doctorat en informatique, Université Paris.

F. Meunier. 1999. Modélisation d’resources linguistiques appliquées à l’industrie. In TALN’99, pages 243-252, Largese, Corse, 12-17 juillet.

A. Nasr. 1996. Un modèle de formulation automatique fondé sur la théorie Sens-texte. Application aux Langues Contrôlées. Thèse de doctorat en informatique, Université Paris.

C. Paris, K. Vander Linden, M. Fisher, A. Hartley, L. Pemberton, R. Power, and D. Scott. 1995. A support tool for writing multilingual instructions. In Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI’95), pages 1398-1404, Montreal.

J. Scheurs and G. Adriaens. 1992. Computers and writing state of the art. Chapter From cogram to alcogram towards a controlled grammar checker. In Proceedings pages 204-221, Kluwer Academic Publishers, London.