Generating a Controlled Language

Laurence Danlos
Université Paris 7
TALANA UFR Linguistique
Case 7003-2, Place Jussieu
75251 Paris, France
danlos@linguist.jussieu.fr

Guy Lapalme
Département d'informatique et RO
Université de Montréal
C.P. 6128, Succ Centre-Ville
Montréal, Québec, Canada, H3C 3J7
lapalme@iro.umontreal.ca

Veronika Lux *
Xerox Research Centre Europe
6, chemin de Maupertuis
38240 Meylan, France
Veronika.Lux@xrce.xerox.com

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, be it human or automatic. A CL consists of a glossary and of 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 Fundamental English* that was elaborated in the sixties (Scheurs and Adriaens, 1992). However CLs are presently being defined 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 the checking tool should not only detect errors but also propose a CL conformable expression. A. Nasr (Nasr, 1996), who worked on the problem of CL reformulation, underlines the difficulties of this task. Reformulation cannot make any hypotheses about the conformity of the input sentences, and therefore must deal with a wider variety of lexico-syntactical constructions than those allowed in a CL. Some instances of noncompliance are relatively easy to detect but much more 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 documentation 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 for generating documentation in 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 Simplified English (SE) (AEC, 1995) and Français Ra-

* Work done while at the Aérospatiale Research Center

tionalisé (FR) (GIFAS, 1996) extend the ATA 100 norms (Bur, 1995) which describe the division of the document into chapters, sections, subsections, etc. reflecting a tree-structured functional organization of the airplane: a chapter corresponds to a system (e.g. main rotor), a section to a subsystem (e.g. gear box), a subsection to a sub-sub-system (e.g. 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 raises the question of the automatic generation of this documentation given some formal representation. This claim led V. Lux (Lux, 1998) to redefine 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 theme according to the ATA 100 norms. It can be divided into smaller modules: for example, the *Task* module is divided into simpler *Sub-Task* modules which are themselves composed of simpler *Instructions* modules. From a linguistic point of view, a textual module uses only a controlled sublanguage. V. Lux thus extended FR to a new CL called FREM (Français Rationalisé Étendu Modulaire) 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 remodularize SE similarly to what has been done to FR with FREM. In this paper, we therefore introduce the notion of an Extended Modular Controlled 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 both to structuration norms 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* components, even though this may be considered as 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, we assume that it is possible to design forms which are sequentially presented to the user to be filled, as in Drafter (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-like structure of the document given in the ATA norms. The goal is that, once the writer has finished filling in these forms, the technical documentation is already properly structured in an abstract language instead of a natural one. In a general text generation setting, using forms to describe what is to be said might seem like a difficult task; but in the context of technical writing, the informational content is almost already prescribed and forms are thus a simple way of complying with the rules of a CL. Indeed in the now common web environments, forms are frequently used for eliciting 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. This can be seen as an illustration of observations made in psycholinguistics. Levelt (Levelt, 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 it **in the way it has** to be written but does not write 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 corresponds to conscious choices. In the NLG scenario, this 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 system has to fill forms but on the other hand, s/he does not have to learn a CL, since compliance with the CL norms is taken care by the *How to say it* component which we now describe.

2.2 *How to say it* component

In this section, it is assumed that if a CL is in fact an EMCL such as FREM, a specific *How to say it* component is designed for each textual module, but always retaining the same formalism.

The lexicon used in the *How to say it* component should be exactly the one enforced by the CL. Similarly, the syntactic constructions and the discourse structures of this component should correspond to the set of allowed constructions / structures in the CL. This can simplify some lexical, syntactic and even discourse choices to be made within the generation 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 constraints on the use of a given lexical item, e.g. in FR a rule forbids the use of *empêcher* (*prevent*) when followed by an infinitive clause. To handle such numerous lexically dependent syntactic rules, a formalism based on a lexicalized grammar is needed. We chose Lexicalized Tree Adjoining Grammar (LTAG) for the following reasons:

- A text generation formalism inspired from LTAG, called G-TAG, has been designed, implemented and used in several applications (Danlos and Meunier, 1996; Meunier, 1997; Danlos, 1998; Meunier and Danlos, 1998; Danlos, 2000). G-TAG takes as input an event graph which can be provided by the user by filling in some forms which ensure that all the necessary information for generation is provided.
- G-TAG deals with textual phenomena such as sentence connectors by extending LTAG to handle discourse comprised of more than one sentence. One of the major innovations of FREM compared to FR (and of EMCL compared to CL) is to implement rules for connecting sentences (clauses). The way to connect sentences has largely been ignored in CLs, although this linguistic issue raises ambiguities which can lead to maintenance errors. For example, simple juxtaposition of sentences is allowed in FR but disallowed in FREM because it is highly dangerous. A technician reading *Nettoyer X. Verser Y sur X.* (*Clean X. Pour Y on X.*) could interpret this to mean either "Clean X with Y" or "Clean X with Z, and next pour Y on X". Only one of these operations is right, the other one may lead to a maintenance error. On the other hand, traditional syntactical ambiguities such as a prepositional attachment will not usually lead to maintenance errors because the technician can usually solve them on the basis of some domain knowledge.
- The lexicalized grammar in G-TAG is compiled from the meta-grammar designed and implemented by M.H. Candito (Candito, 1996). This makes it easy to follow the evolution of rules of 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 disallowed (in a 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 4. It is written in Flaubert, an implementation of G-TAG

(Danlos and Meunier, 1996). The *How to say it* 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 fine tune the final text.

So, automatically generating technical documentation seems technically possible provided the technical writer is willing to fill forms which in principle should be less demanding than learning the rules of an (EM)CL. This approach also has other 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 texts in several languages, and to regenerate updates as often as necessary, 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 for this reason that FR was developed by adapting SE, in order to ease the translation from French to English. FR authors try to ensure that everything that can also be written in FR can be translated into SE. From this point of view, the definition of a source CL_1, depends on the definition of a target CL_2. Developers of CL_1 are more likely to select structures which can be easily or even literally translated into CL_2. What then happens if CL_1 and CL_2 are structurally different? This can lead to a situation where CL_1 imposes a cumbersome writing style that contravene conventions shared by native speakers of L_1, thereby contradicting CLs' aim of enhancing understandability. Rules of an (EM)CL should be elaborated without such multilingual considerations. Their definition should principally pay attention to the characteristics of one language, trying to avoid typical ambiguities. Such criteria are difficult enough to deal within a single language without taking translation problems into account.

Now if we consider multilingual generation in (EM)CLs, we find that there are major benefits from the multilingualism modeling proposed by NLG. In particular, defining a common representation is possible since the structure of the documentation is language independent. Recall from section 1 that the thematic structure of the documentation in the aeronautical domain must reflect the functional decomposition of the airplane and that the same textual modules can be identified in many languages. Thus 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 it* component (Section 2.2) need be adapted to the target (EM)CL which should be monolingually defined.

3.2 NLG as an aid for testing and developing a CL

An NLG system can provide concrete assistance for the 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 an illustration, a major difficulty in CL definition concerns the coherence between the lexicon and the writing rules, as illustrated by (Emorine, 1994) with the following example:

- *Empêcher l'oxygène de s'accumuler* (*Prevent the oxygen from accumulating*) does not conform to a FR lexically dependent syntactic rule, according to which *empêcher* (*prevent*) should not be followed by an infinitive clause.
- *Empêcher l'accumulation d'oxygène* (*Prevent oxygen accumulation*) does not conform to FR lexicon, according to which the verb *s'accumuler* (*accumulate*) should be used instead of the noun *accumulation* (*accumulation*)
- *Empêcher 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 if 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 be used for checking a CL, which is helpful even if the CL is intended for a human writer because it may avoid the discovery of various cases of incoherence by the writer. If the writers can justify their writing difficulties by pointing out inconsistencies in the CL definition, they won't be motivated to use what they will tend to consider as an absurd invention by people who understand nothing about the job.

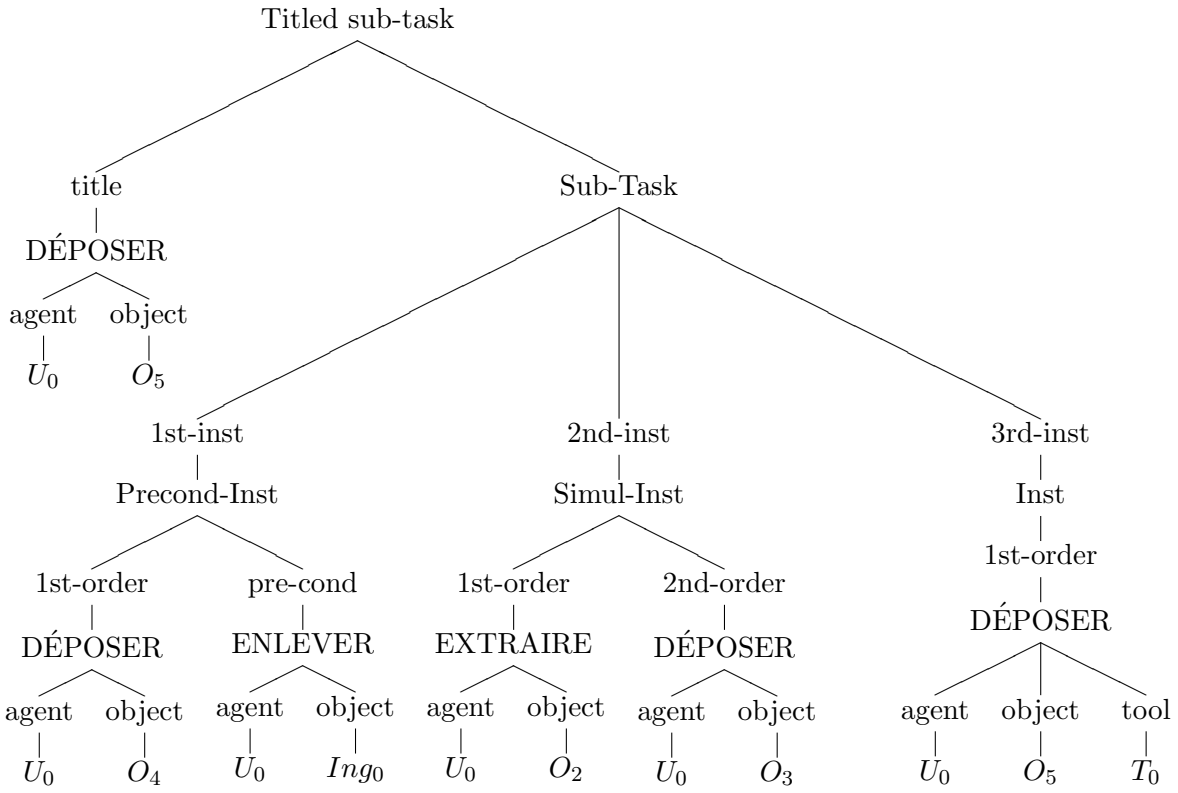

Figure 1: Event graph given as input to Flaubert. In the prototype, this information is entered in textual form.

NLG can also help strengthen CLs' claim to lead to more homogeneous texts, which is equivalent to forbidding certain paraphrases. NLG precisely deals with paraphrase as, for some inputs, a NLG system will produce several texts. In this way, NLG helps identify which paraphrases still remain possible in the CL. In practice, when an NLG system proposes 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 some 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 to our knowledge had never been examined in such details. To further pursue, V. Lux (Lux, 1998) has developed a proof of concept generator using Flaubert (Meunier, 1997; Meunier and Danlos, 1998) to see how these theoretical concerns could be applied in practice. The generator can produce text for about ten subtasks in FREM. These tasks comprise from two to eleven instructions, illustrating ten different instruction types such as: simple instruction with a goal, simple instruction with a condition, complex instruction with simultaneous actions, etc. They involve the use of various syntactical constructions such as infinitive or sentential subordinates, nominalisation, negation, etc.

Input to the prototype are event graphs such as the one given in Figure 1. The output is a well formed French text such as the one in Figure 2 which was generated from Figure 1. In Lux's prototype, the event graphs were hand coded, but now Flaubert has been rewritten in CLEF (Meunier, 1999; Meunier and Reyes, 1999), which has a better graphical input mechanism that would have eased the input process.

The output text is a sub-task including a title and instructions of different types (only the first three instructions are given in the Figures) to be performed by the same person (e.g. U_0). FREM defines which connector to use for each instruction type (e.g. conjunction *et* for an instruction with simultaneous actions).

The generation of noun groups for the objects (O_i), ingredients (Ing_i) and tools (T_i) re-

Sous-tâche 60-007
3.1 Dépose du segment d'arrêt (5)
- Après avoir enlevé le mastic PR, déposer le segment d'arrêt (4).
- Extraire le porte joint (2) et déposer le joint (3).
- Déposer le segment d'arrêt (5) à l'aide de l'outillage (Z).

Figure 2: Text generated by Flaubert from the input of Figure 1

lies on a mapping table between these labels and their denominations; this was a temporary solution for problems outside the scope of the prototype. We should have relied on existing nomenclatures for tools and ingredients, and on the fact that objects are systematically represented in drawings associated with various subtasks e.g. O_5, called *segment d'arrêt*, is labeled (5) on the drawing associated with the example above. In a graphical interface environment, authors would select these objects linked to a controlled terminology data base.

This proof of concept generator served well our purpose of testing our theoretical ideas but unfortunately it could not be evaluated in a realistic CL text production environment. Our sponsors were very interested in the results we have produced but changes in their organisation made it impossible to carry further investigations. We intend to further pursue our research and use the new implementation of Flaubert to generate controlled language in an other area of application while keeping the concept of an extended modular CL.

5 Conclusion

This paper has argued that linguistic norms imposed by CLs should not be considered in isolation. They are only a part of a set of more comprehensive norms on the document structure and layout. This insight led us to define a notion of textual modules, each with its own linguistic norms, and to envisage the generation of technical documentation using an extended modular controlled language (EMCL). Norms for document structure such as ATA100, its linguistic characteristics and its layout requirements may be seen to respectively define the text structuring, the linguistic and the layout components of an NLG system.

We have also shown that a generation point of view can help refine the definition of an EMCL. The EMCL can be defined monolingually, multilinguality being obtained through NLG. These ideas were tested within a proof of concept text generator 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 Macklowitch who suggested many improvements to the paper.

References

AECMA Document PSC-85-16598, 1995. *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. *Spécification ATA no 100, traduction française. Specification for Manufacturer's Technical Data - ATA Specification 10*, 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 Applications (CLAW)*, Leuven.

L. Danlos and F. Meunier. 1996. G-TAG, un formalisme pour la génération de texte : présentation et applications industrielles. In *Actes du colloque Informatique et Langue Naturelle*, Nantes.

L. Danlos. 1998. G-TAG: un formalisme lexicalisé de génération de textes inspiré de TAG. *Traitement Automatique des Langues - T.A.L.*, 39(2):4–32.

L. Danlos, 2000. *Tag Grammars*, chapter G-TAG: A Lexicalized Formalism for Text Generation inspired by Tree Adjoining Grammar. CSLI.

M. Emorine. 1994. Projet de recherche sur la modélisation des entrées verbales du français

rationalisé. Technical report, Université de Clermont II.

GIFAS. 1996. Guide du rédacteur - partie 2: Français rationalisé. Technical report, GIFAS, Paris.

A. Hartley and C. Paris, 1996. *Le texte procédural : langage, action et cognition*, chapter Une analyse fonctionnelle de textes procéduraux : apport de la génération automatique à la définition des langues rationalisées, pages 211–222. Toulouse.

W. Levelt. 1989. *Speaking - from intention to articulation*. MIT Press, Cambridge Massachusetts.

V. Lux. 1998. *Elaboration d'un français rationalisé étendu pour un manuel de maintenance aéronautique, test en génération automatique*. Thèse de doctorat en linguistique, Université Paris 7.

F. Meunier and L. Danlos. 1998. FLAUBERT: an user-friendly system for multilingual text generation. In *Proceedings of the 9th 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, formalisme pour la génération inspirée des grammaires d'arbres adjoints*. Thèse de doctorat en informatique, Université Paris 7.

F. Meunier. 1999. Modélisation des ressources linguistiques d'une application industrielle. In *TALN'99*, pages 243–252, Cargèse, Corse, 12-17 juillet.

A. Nasr. 1996. *Un modèle de reformulation automatique fondé sur la théorie Sens-Texte - application aux Langues Controlées*. Ph.D. thesis, Université Paris 7.

C. Paris, K. Vander Linden, M. Fischer, A. Hartley, L. Pemberton, R. Power, and D. Scott. 1995. A support tool for writing multilingaul instructions. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI'95)*, pages 1398–1404, Montréal.

J. Scheurs and G. Adriaens, 1992. *Computers and writing - state of the art*, chapter From cogram to alcogram : toward a controlled english grammar checker, pages 206–221. Kluwer Academic Publishers, London.