[*Machine Translation Today: Translating and the Computer 15*. Papers at a conference…18-19 November 1993 (London: Aslib)]

SIMPLIFIED ENGLISH GRAMMAR AND STYLE CORRECTION IN AN MT FRAMEWORK: THE LRE SECC PROJECT

G. Adriaens[1]
Siemens Nixdorf Software Centre Liege, Rue des Fories 2, B-4020 Liege
University of Leuven Centre for Computational Linguistics, Maria-Theresiastraat 21, B-3000 Leuven
GeertAdriaens@csl.sni.be

This paper describes ongoing developments in the LRE-2 project SECC (A Simplified English Grammar and Style Checker/Corrector). After a general description of the project, the approach to building the SECC writing tool is discussed. First, lingware issues are dealt with: resources used, technical implications of simplified grammar correction as machine translation, testing and evaluation issues. Next, we take a look at software issues, in particular the user interfaces. Finally, we discuss some open issues and future developments.

GENERAL DESCRIPTION OF SECC

SECC is an LRE-2 project that started in November 93 and runs until May 96 (2.5 years). Partners in the project are Siemens Nixdorf (B), Cap Gemini (F), Alcatel Bell (B), the University of Leuven (B), and Sietec (D); its total foreseen effort is about 13 person-years. SECC's main goal is the development of a tool for technical writers who produce documents in a variant of Simplified English (SE) described below. The tool will check if the documents comply with the syntactic and lexical rules; if not, error messages are given, and automatic correction ("translation") is attempted wherever possible to reduce the amount of human correction needed. Special attention will also be paid to non-native writers of English (French, Flemish, and German writers): it is useless to check SE rules if rules of English in general are violated in the first place. This will add an extra level of complexity to the tool, as will be discussed below. Further attention points of SECC include syntactic checking and correcting at textual levels beyond the sentence (from paragraph to full text), and full integration within a DTP environment (InterleafS). In all these developments SGML and its associated tools play a major role.

GRAMMAR CORRECTION AS MACHINE TRANSLATION

One of the basic ideas of the SECC project is to treat SE grammar checking and correcting as a problem of translating English into SE. Given an MT system and its development environment, the belief is that the machinery offered should be sufficient to create a special language pair English-SE. At the same time, a development like SECC is meant to show that NLP components and environments designed for MT in the first place can be (re)used for other applications. For SECC, we are using the transfer-based Metal® MT system and its development environment. As will be explained below, the analysis-transfer-generation cycle is mapped to an analysis-diagnosis-correction cycle.

[1] SECC is also: Lieve Macken, Luc Pauwels (both University of Leuven), Anne Derain (Cap Gemini), Frederik Durant (Siemens Nixdorf), Patrick Goyvaerts (Alcatel Bell), and Uus Knops (Sietec) whose teamwork has formed the basis of this paper. Gert De Braekeleer and Bart Depoortere (both ex-Siemens Nixdorf) also helped lay the foundations of SECC.

Resources Organisation

In a regular Metal language pair, the following lingware components are relevant:

- source lexicon
- analysis grammar
- transfer lexicon
- transfer grammar
- target lexicon
- generation grammar

analysis
|
transfer
|
generation

Analysis. For the English-SE language pair, the source language remains a regular language. Hence, the existing English lexicon of Metal (over 50000 base forms) is reused as source lexicon, as well as the existing English analysis grammar. Whereas the lexicon can be taken over as such, this does not hold for the analysis. The input the Metal analysis grammar handles is in principle only correct regular English, but robustness has required that some semi-grammatical or even ungrammatical structures be accepted by the parser. These include phenomena like sloppy punctuation or uncommon adverb placement These characteristics of the English analysis are important in the context of the non-native writer support SECC wants to offer in addition to regular SE checking. Non-native writers make mistakes that sometimes render a sentence ungrammatical, and hence make the analysis or any further processing behave unpredictably or go totally wrong.[2] Non-native writer support requires that the analysis grammar be extended to deal with this kind of mistakes. Some of those fall into the category of the phenomena already handled for robustness' sake, but others definitely do not. We will come back to the technical details of the approach planned in SECC in the section on open issues and future work.

Transfer. As with any new language pair in Metal, it is the transfer part that must be newly developed. The transfer lexicon takes care of lexical mappings between source and target, and the transfer grammar handles structural mappings. Given the particular nature of the target language, we will first take a look at the definition of SE[3] as used in the SECC context SECC's SE represents a subset of regular English, consisting of Alcatel Bell's COLEX (a restricted regular English vocabulary), COTECH (a restricted technical English vocabulary from the domain of telephony) and COGRAM (a restricted grammar).

COLEX contains about 1500 accepted regular English words. It was built both from existing SE lexicons (to the extent that they were accessible) and from a word frequency list at Alcatel Bell compiled from a large body of in-house telecommunication texts. COLEX also defines some 500 "translations" of regular English words into one of the 1500 accepted SE words. These translations are implemented in the transfer lexicon of the language pair at hand. A few examples (adjective, noun and verb respectively):

rapid --> fast
quick --> fast
prompt --> fast
swift --> fast

category --> type
nature --> type
variety --> type
class --> type
kind --> type
model --> type
sort --> type

[2] When no overall sentence analysis is found, the Metal chart parser returns a so-called phrasal analysis, a collection of the largest phrases identified in the sentence. These "chunks" can then still undergo useful further processing (cp. Critique's fitted parse, Ravin (6)).

[3] We will not go into the different ways one can define Simplified English. See on this and other SE-related matters Wojcik et al (11, 12), van der Steen and Dijenborgh (8); Adriaens and Schreurs (1); Schreurs and Adriaens (7); Humphreys (4), Pulman and Rayner (5).

alter --> change
modify --> change
convert --> change (as in convert money)
transform --> change

The examples are a simplification of the status of the respective words in COLEX, in a sense that the precise contexts of application are left out. Also, non-SE words may have different translations, depending on these contexts. *Prompt*, for instance, also has a translation into *immediate*, *category* also goes to *group*, and *convert* to *adapt* (as in *convert a building*). We will come back to this in the technical details below. COLEX is not definitive, given that changes are possible at the request of users or as a consequence of computational problems in the process of building the tool (lack of precision in the paper version, inconsistencies, circularities, etc.).

COTECH is still in the development stage, and is currently a collection of different technical terminology bases at Alcatel Bell. One important subset of particular interest to SECC is the technical terminology from a 2000-sentence corpus that SECC uses as test material (see the subsection on evaluation below).

COGRAM is a 150-rule SE grammar, again based on available SE rule material and (more importantly) on problems found in the test corpus. About two-thirds of these rules are computationally tractable in the SECC context (i.e. either to do diagnosis or to do diagnosis plus correction). The grammar is organised into four major categories; we give an example rule for each category:

Textual Control

(1) Do not use articles in titles or headings consisting of a noun or noun cluster.

Syntactic Control

(2) Do not express an idea in parentheses or between dashes inside a sentence. Use a separate sentence instead.

Lexical Control

(3) Use short regular action verbs. Avoid commonplace verbs such as ***do, make, get, be, perform*** *combined with action nouns.*

Character and Punctuation Control

(4) Do not use an apostrophe in expressions such as ***in the 1970s, during the 80s, into the 90s.***

Most of the rules are formulated with "Use (only) X", "Do not use Y" or "Avoid Z". "Do not use Y" implies: Y is always an error, "Avoid Z" implies: if you cannot but use Z, then it is acceptable. With an eye to implementation of these rules in SECC's transfer and generation grammar (for their diagnosis and correction aspects respectively), it is important to add that "Do not use" or "Avoid" rules are as much as possible complemented with a "Use (instead)" part. If this is not the case, the reason is either that it is obvious from the wording (as in rules (1) and (4) above), or that the correction complement of the phenomenon is too complex. All rules are further accompanied by one or more examples (i.e. pairs of wrong and correct sentences) taken from the Alcatel Bell SECC test corpus. In case the correction complement is too complex, the examples suggest typical cases and how they can be remedied. Beside being a pedagogical requirement, the "Use (instead)" part is a computational requirement for doing automatic correction.

Generation. Once the transfer phase has annotated the analysis tree with all diagnosis information about detected errors, this information can be turned into system output in a generation phase. For an SE checker without correction, this output consists of error messages (critiques).[4] In the MT context, *generation* normally has a stricter meaning: the input source string is replaced by a target string. In SECC, both types of generation are done. If correction is possible, part of the SECC output will be an SE equivalent of the erroneous regular

[4] In the Metal context, grammar and style checking experiments (without correction) of this kind had already been done for German before SECC in the context of the TWB Esprit project (see Thurmair (9,10)).

English input sentence. (It is only when such source to target string mappings take place, that we speak about correction.) Moreover, both in cases where correction is possible and in cases where it is not, SECC will give error messages about the input. A distinction we make in the context of correcting or not is that between a *possible* and a *certain* error (corresponding to *weak* and *strong* diagnosis respectively). Weak diagnosis (possible error) corresponds to cases where a problematic construction can be identified, but not diagnosed to be wrong. Two examples: *Avoid splitting infinitives, unless the emphasis is on the adverb*; *Do not use* when *in conditional clauses*.[5] For the first rule, a split infinitive can easily be detected in Metal, but emphasis cannot; for the second, when can be detected, but we have no waterproof way to distinguish temporal and conditional meanings. Strong diagnosis (certain error) corresponds to cases where the system can safely assume there is an error (see the example rule below). In case of weak diagnosis, we do not correct the sentence automatically. Whether all cases of strong diagnosis can (should) be corrected automatically is an open issue, given that we are still in the process of implementing correction. In any case, if correction is not done, SECC will still make useful suggestions about how to change or rephrase the sentence. How SECC produces its different output types is discussed in the next section.

To come back to the resources needed for generation, we have to say something about the generation grammar and lexicon.

About the generation grammar, we can be brief: it uses the available Metal generation machinery (feature percolation, tree transformation, string generation), and reuses (small) parts of the existing English generation component (as used in German-English or French-English). Given that the target is a subset of the source, this generation component is not as extensive as in a regular language pair. For one thing, not all input sentences contain errors; if so, they remain unchanged in the output. For another, a fair amount of errors can only be diagnosed, but not corrected; here too, no changes are made. And finally, even if corrections are made, they are often local in nature, so that parts of the input can simply be taken over in the output Still, a non-trivial issue in generation is that of multiple generation in case SECC suggests different alternatives for correction; we will discuss this issue in the section on the technical approach. An example correction at the noun phrase level is the following:

Given the rule

Use of *in the genitive case when a possessive noun form is inanimate*,

a noun phrase like *The module's most important function* is corrected to *The most important function of the* module.

The target lexicon deserves a little more attention, because it is treated in a special way in SECC. In principle, we could create a monolingual dictionary that contains all SE-accepted entries. This approach is not taken. A regular lexicon in Metal is used both for analysis and for generation: the same English lexicon serves both in English-German and in German-English. Hence, the existing English entries contain all information needed for analysis and generation. Creating a monolingual SE lexicon would lead to duplication of information (present in both the English and the SE lexicons) and to difficulties in maintainability. For certain types of information (e.g. morphology), a change in one of the two lexicons would entail the necessity to change it in the other. A second approach could be to mark all SE words as such in the existing English lexicon (and have only one lexicon loaded). This approach also creates problems, both conceptually and computationally. First one must know that Metal does not have semantic reading distinctions in its monolingual dictionaries; semantic distinctions are only made in transfer. So, one can hardly mark a word as SE, if only one of its meanings is actually SE. Second, the English lexicon serves in many language pairs. If for one such pair, a word needs to be added, changed or removed, the effects for SECC are unpredictable (and vice versa). Moreover, marking words for a particular application reduces overall reusability and portability. Then what do we do? As the treatment of reading distinctions already suggests, the transfer dictionary is the place to be. The solution opted for is to leave the English lexicon untouched (using it both for analysis and generation), and move the information about the SE nature of words into the English-SE transfer lexicon. The rule is simple: if a word X is SE, then there must be a reflexive transfer entry X -- > X. For the examples given in the previous subsection, the transfer lexicon must contain

[5] The actual COGRAM rule is more elaborate than this; matters are simplified here.

fast -- >fast
type --> type
change -- > change
immediate -- > immediate
group -- > group
adapt -- > adapt.

Depending on whether a word is never SE, or is only SE in one of its meanings, the following entry configurations occur in the transfer lexicon. We use as a convention that symbols for non-SE words are in lowercase, those for SE words are in uppercase. We are also only concerned with the behaviour of a word within one word class.

1. A word *x* is never SE; it has *Y* as its SE equivalent

x --> Y	e.g.	altitude --> height
Y-->Y		height -- > height

There may be other entries *x* --> *Y* (if some other word also translates into *height*), but not the entry *x* --> *x* (*altitude* -- > **altitude**).

2. A word *x* is SE in one meaning, but not in another. (A pair *x/X* represents the same word, with *x* its non-SE meaning(s), *X* its SE meaning.)

x -- > Y e.g.	since -- > because *if subclause of reason*
X -- > X	since --> since *if subclause of time*
Y -- > Y	because --> because

Another example for nouns is the set of mappings around *error* (we leave out the contextual requirements for the different translations):

error -- > defect	|
error -- > fault	| *error* is never SE
error -- > mistake	|

defect -- > defect	
fault -- > fault	
mistake -- > mistake	| *mistake* is sometimes SE
mistake -- > defect	|

The construction of this kind of lexicon is not an easy task. For one thing, the amount of semantic information available in the Metal system is not always sufficient to make the subtle reading distinctions intended. This will have implications on the kind of lexical error correction SECC can offer. As with the regular language pairs, in case the system cannot decide which meaning is intended, it offers all possible translations. We will come back to this in the next section. As a final note concerning the lexicon, let us add that the Metal system provides a wide range of tools to consult and maintain its lexicons. For example, we regularly run consistency checks to see if

- all source entries in the transfer lexicon exist in the English lexicon
- all target entries in the transfer lexicon exist in the SE lexicon, i.e. are SE-accepted (by checking if there is a reflexive transfer for them in the transfer lexicon)
- no circularities occur in the transfer lexicon (i.e. there must not be both an entry x -- > Y and y -- > X).

We also hope to be able to refine the lexical relationships between regular English and SE by extracting the "entry webs" that are created as coding continues (see the error case above).

Technical Approach

Given the different resources sketched in the previous section, we will now have a closer look at a few interesting implementation details.

The peculiar nature of the Englisb-SE language pair is a challenge for Metal in many ways. A requirement that has influenced the organisation of the analysis, transfer and generation phases is the complex nature of the SECC output. In the previous section, we mentioned that SECC outputs diagnosis information about the original input sentence (with suggestions for correction or improvement) as well as corrected output sentences (whenever possible); it combines the output of a grammar checker and an MT system. Given the MT context of SECC, the only output the system gives in principle is the target string. Moreover, Metal works in such a way that the input string is gradually overwritten with the output string as a result of the manipulations of the tree constructed on top of this string. Recovering the input string (let alone a diagnosis-annotated input string) from the final tree is nearly impossible. In short, the classical analysis-transfer-generation cycle has required some adaptations for SECC.

Analysis. As to the analysis phase, the problem of the disappearing input sentence already starts here. Let us take the example sentence *The module's major function is to localise the fault.* At the end of the English analysis, the string that forms the leaves of the analysis tree is *The module's major function is function to localise the fault.* In other words, the analysis creates a kind of deep structure, with the implicit subject of the infinitive clause made explicit. In this case, material is added that did not occur in the input string. In other cases, material may be moved (adverbs, for instance), or even deleted (inflections and auxiliaries, for instance, are featurised). The reasons for these manipulations can be found in the particular MT application: already during analysis, the input string (or rather, the tree dominating it) is "prepared" for translation into a target language. To overcome this problem, SECC adds a post-analysis phase which adapts the analysis tree so that its leaves correspond to the original input sentence again, while leaving all vital analysis information intact To this end, information is retrieved from Metal's chart parser structures (which contain a complete record of all manipulations of the input).

Transfer. Once this analysis tree is re-constructed, it can enter the SECC transfer phase. Here, the lingware routines corresponding to the COGRAM rules inspect the tree for feature information or particular clause patterns, annotating it with error labels whenever necessary. At the same time, the transfer lexicon (COLEX/COTECH) is consulted to retrieve SE equivalents of the non-SE words, together with the information as to how they must appear in the corrected output All this transfer information is stored in the original analysis tree without changing its structure or replacing its leaves. Here again, it is the necessity to annotate an untouched input string that requires transfer to be structure and string preserving. All structural and lexical changes to the output are postponed until generation. We stress this aspect because it is not the way Metal works for regular language pairs. There, transfer and generation can work in an interleaved fashion, with lexical substitutions and tree transformations gradually overwriting the analysis tree and the input string (because no longer needed in the end).

Generation. As to generation, we already mentioned that SECC should generate two kinds of information: an output string (when correction can be done), as well as useful diagnosis information plus suggestions for correction or improvement. Still, we want to respect the Metal generation mechanism, simply creating an output string for each input sentence, without side-effecting messages along the way. One special reason for going through the regular string generation is to keep taking advantage of the layout-preserving features of Metal, not only on the text level but also on the sentence level (preservation of change in character styles — font size, bold/italic, etc.). In principle, a user can then run SECC, and simply get an output text with all corrected sentences "inserted" into his original input text without loss of layout information. How can these requirements be fulfilled? In the first place, we designed an SGML output representation. It is a complex string object annotated with SGML tags. Simplifying matters a little, it has two major elements, corresponding to the two types of generation SECC must perform, a CORR(ection) and a DIAG(nosis) element The CORR element contains the corrected input sentence, or the original input sentence itself if nothing was found wrong or if correction was not possible. The DIAG element is more complex: it repeats the input sentence, with non-crossing error tags generated around the linguistic entity they apply to. An example can make this more concrete:

The module's major function is to localise the fault

```
<CORR>
The <ERR-W64 ALT=("primary")>most important </ERR-W64> function of the module is to locate the fault.
</CORR>
<DIAG>
 <ERR-S30>
   <ERR-WG78>
      the module's
        <ERR-W64 COR=("important" "primary")>
             major
        </ERR-W64>
            function
    </ERR-WG78>
     is to
        <ERR-W64 COR=("locate")>
          localise
        </ERR-W64>
     the fault
  </ERR-S30>
</DIAG>
```

ERR-S30: *End a sentence with a full stop.*
ERR-WG78: *Use* of *in the genitive case when a possessive noun form is inanimate.*
ERR-W64: *Use only SE-accepted words.*

(Note the correction of *major* to the superlative of *important*, as well as the preservation of the boldface in the corrected noun phrase.)

This object contains everything needed to present any part of its information (in particular to a user, see below) in whatever way one chooses, given SGML support tools. Storing the object in a structured database (for instance, to use it for evaluation purposes) is also straightforward.

How does Metal generate such a complex string instead of a "simple" target sentence? There are two generation runs, a first one for the diagnosis information, and a second one for the corrected output sentence. The first run recursively descends the tree, and splits up the error identifiers put on the nodes during transfer into an opening and a closing tag. These tags are wrapped around the node, and treated as "words" of the target language, just like the actual words of the input sentence. The first run collects all these "words", and puts the resulting string on the root node (S) in a feature-value pair. The second run is a regular Metal generation run finally transforming the transfer tree and the input string into a corrected SE string. At the end of this double generation cycle, both strings are concatenated, giving the final translation output.

To conclude this discussion about the technical details of the analysis-transfer-generation cycle of SECC, a word is in order about possible alternatives in the corrected output sentence. This will also explain the appearance of an ALT (alternatives) attribute in the corrected output string (see the example above). The transfer lexicon examples given in the previous section have shown that sometimes alternative SE translations exist for a word. Given the SE principle "one word, one meaning", this should only be the case if these alternatives represent distinct meanings of their English counterpart. The simple feature semantics in Metal does not always permit to encode these distinctions computationally; Metal's translation process is also non-interactive, so the user is not asked for help during the translation. The way this problem is handled is by simply offering the alternatives in the output string. One alternative (the most frequently used one in the domain at hand, for instance) goes through the process of full sentence generation, and appears correctly integrated in the sentence (correct inflection, correct surface form of surrounding elements (a/an, for instance), correct distribution over the sentence for multiword entries, etc.); the others are attached to it as possible alternatives. Hence the ALT attribute in the corrected output sentence. (Whether one of the alternatives is chosen instead of the proposed one depends on the end user of the system.) Whereas in regular Metal language pairs these alternatives are base forms, in SECC the words undergo morphological generation, which reduces the amount of postediting needed when an alternative is chosen. Because the attachment of this kind of alternatives creates a grey area between real correction (in the strict MT sense defined above) on the one hand and mere suggestions on the other, we have considered multiple generation. This would mean that for each

alternative (or for each possible combination of alternatives – a sentence can contain more than one choice point), the full generation cycle is gone through. The CORR output part would then be a set of fully generated sentences, with all alternatives worked out. This issue is currently under investigation, given the non-trivial changes it requires in the generation algorithm and the potential negative effects on system performance. Currently, we also do not have a clear idea of the amount of variation that will be generated in the corrections. If we respect the nature of SE, this amount should be kept to a minimum, even if the system is not powerful enough to deal with all semantic and pragmatic subtleties. Finally, multiple generation would require a different (more complicated?) user presentation than the one foreseen now.

Evaluation Issues

As we already mentioned in the general project description, SECC's target domain is that of telephony, with Alcatel Bell as provider of resources in this domain. One of these resources is a 2000-sentence corpus, which we annotated manually for all SE problems. It has already formed an invaluable source of information to tune COGRAM and illustrate it with real-life examples for the intended users. Once the major parts of COGRAM, COLEX and COTECH will be implemented, the corpus will play a central role in system tests. The question then arises how the SECC output will be evaluated. In Metal's current testing and evaluating environment Sisyphus (see Adriaens et al (2)), regular translation output is rated by human evaluators as to correctness or understandability. Output quality then receives a score based on the amount of correct, wrong and understandable translations. For SECC, the complexity of the output does not allow "simple" judgements. Still, we can say something about the evaluation criteria we intend to use.[6] We also refer to Wojcik et al (12) for a report on the evaluation of the Boeing Simplified English Checker, where comparable problems and criteria are discussed in detail.

In the context of information retrieval and error treatment in computer programs, the notions of precision and recall (associated with the complementary notions of noise and silence, or overkill and undershoot) are used for evaluation. In the context of grammar checking, they refer to the following rates:

$$\text{Precision} = \frac{\text{Number of correctly flagged errors}}{\text{Total number of errors flagged}}$$

$$\text{Recall} = \frac{\text{Number of correctly flagged errors}}{\text{Total number of errors actually occurring}}$$

Good precision requires a low rate of spurious errors (noise, overkill); good recall requires a low rate of missed errors (silence, undershoot). To give an example (the only one currently available for grammar checkers): Wojcik et al (12) report a precision rate of about 80% (20% noise), and a recall rate of about 90% (10% silence) for the tests they ran with their Boeing SEC. For the SECC prototype, we are aiming at a precision rate of 75% and a recall rate of 80%, although these figures are quite arbitrary given the non-availability of sufficient comparative material. Moreover, all figures are oversimplifications because they treat errors on a par; more refined calculations should take into account the relative weight of errors.

Another rate which we think is interesting to consider in evaluation of grammar checkers (and especially correctors) is what could be called the convergence rate. For SECC, it is related to what happens when the tool is applied to a text in different cycles. Suppose it is run on an input text, producing a number of automatically corrected sentences.[7] These sentences are then resubmitted to the same version of the tool. In principle, this second run (or any subsequent run, for that matter) should leave them untouched. Otherwise, there are problems with the application of the grammar rules (sloppy correction, rules feeding each other, etc.). For the SECC prototype we (again arbitrarily) set the required convergence rate at 80%: at most two sentences out of ten automatically corrected ones may generate error messages.

[6] For evaluation of SECC, we will closely collaborate with the TSNLP LRE project (Test Suites for Natural Language Products). TSNLP will use COGRAM and COLEX as a basis for constructing test suites for controlled language checkers like SECC.

[7] For a checker not doing correction, the convergence test could be applied to human corrections of a text. For SECC, this could also be done, but then a distinction should be made between human and computer corrections.

SECC USER INTERFACES

In the preceding sections we have concentrated on the sentence-level lingware (grammar and lexicon) activity of SECC. An important part of the software work concerns the creation of the user interfaces (in Motif on Unix platforms). Due to space limitations, we cannot include screen dumps of the main SECC windows; we will briefly give a very general description of the functionality of the different interfaces. In this section, the batch interfaces are described; in the next section, the future interactive interface is touched upon. As a terminological note, the distinction batch-interactive refers to the way a request for a SECC run is handled. In batch mode, a complete file is submitted to the SECC processor, queued, processed, and file output is sent back at a later stage. In interactive mode, a document fragment selected inside a text processor is sent directly to the SECC processor, and an output result is expected within a few seconds; from the user point of view, a good response time is crucial. The reason for stressing this technical distinction is that interactive can also be used to refer to the interaction between the computer and the user. Even in a batch system, a lot of this interaction can take place. For SECC, this is certainly the case, as will be described below: the user can walk through the results of the batch process in a way that implies a fair amount of system-user interaction.

A first interface (the *independent batch interface*) allows access to SECC outside of any text editor or desktop publishing package. It is an extension to the existing Metal interface, and mainly aimed at batch processing of input documents originating from different text processing systems (Word, WordPerfect, FrameMaker, Interleaf, Ventura,...). Metal separates the text from the layout information, and the text is run through SECC in batch mode. Two output files are produced: an optional one with just the results of correction (taking over the correct input sentences, and replacing the erroneous ones with the proposed correction), and an error report file with fully SGML-tagged sentences (including SECC statistics at the end). If the user chooses to, he can post-edit this file by consulting the diagnosis information and modifying the proposals for correction in a simple editor. Eventually, the correction part of this file (untouched or post-edited) is restored in its original layout. Additional functionality offered is related to setting parameters of SECC (level of expertise, activation of non-native support, etc.), choosing on what server to execute the SECC run, and manipulating the queue of SECC jobs.

Whereas the Metal system is normally an independent application that does not run inside a text processing system, one of the requirements for SECC is that it does run inside such a system, namely Interleaf. This second SECC interface is the integrated batch interface. Interleaf was chosen because it is the company-wide standard of the intended user in the SECC consortium (Alcatel Bell), and also because it offers good tools for integrating applications into it. In the meantime, Interleaf6 also has become Motif-based, and it supports SGML. In order to meet the Interleaf integration requirement, a Metal Application Programming Interface was first constructed so that Metal functionality is accessible to other applications. Using this API[8], all the functionality described in the previous paragraph for the independent batch interface is also implemented inside Interleaf, in addition, it offers more bells and whistles as to the presentation of error messages, pasting corrections into the document setting SECC parameters, etc. Finally, users can suggest changes to COLEX or COTECH via a coding interface. Note, by the way, that given the nature of SE, lexical changes cannot be made freely, but need to go through a lexicon administrator.

As a general characteristic, the SECC interfaces are text-driven (like most other SE checkers): error messages and corrections are stored with each sentence object and are normally accessed via that object. Another approach one can take to the presentation of the tool's results is a grammar-driven one. Here, the applied rules determine the view one gets of the results, and actions are associated with a rule or rule class (retrieving all sentences to which the rule applies, for instance).[9] Both views are complementary, but in the current planning of SECC we will concentrate on the text-driven approach.

[8] Although we could have accessed Metal functionality directly in the independent batch interface, we also used the API to build it, because it requires less system-dependent programming.

[9] See e.g. the checklist facility in Oracle's CoAutbor (a commercial SE checker), or the more recent SE checker developed by GSI-Erli in the context of the GRAAL project

OPEN ISSUES AND FUTURE WORK

An important issue we have not gone into yet (also because we cannot report on implementation results) is the non-native support SECC wants to offer. As mentioned before, in a multilingual context where English plays an important role many non-native writers produce technical documents in English or SE. As earlier tests have shown (see Adriaens & Schreurs (1)), the majority of mistakes made are due to interference of the native language and do not concern SE in the first place. However, if nothing is done about them, there is not much use to an SE checker either. SECC will try to capture some of the most frequent grammatical and lexical mistakes made by Flemish[10], French and German writers. From the three studies we did on this subject, the following sentences contain some typical examples:

Flemish writer

Eventually, another telephone may work *perfect* too. (*Eventually*: Dutch *eventueel* = English *possible*; should be *Possibly/Maybe*; *perfect*: Dutch adverbs have the same form as their counterpart adjectives; should be *perfectly*.)

French writer

It is a well *argumented* text in which *is discussed the problem of system maintenance*. (*argument* < French *argumenter*, English = *argue*; wrong (French) subclause inversion)

German writer

Our system is better *as* the one *who* is offered by the competition. (German *besser als*, English *than*; German classification of relative pronouns is gender-based, not sex-based, hence confusion *who/that*)

We are fully aware of the additional complexity this adds to the system. Up to now, SECC did not need to intervene during the analysis phase: the input is expected to be regular English. Given non-native mistakes, the analysis phase in itself will need a recovery or correction component to obtain an analysis that is usable by the COGRAM rules in transfer. The techniques we are investigating here are rule condition relaxation (see e.g. Ravin (6), or Alonso (3) for Spanish in the Metal context) as well as the addition of low-level fallback rules to be applied if the regular grammar rules fail. In any case, because of potential unpredictable interference effects inside the grammar, the non-native check will be a feature that can at all times be switched off by the user.

All the work in progress reported in the preceding sections actually relates to phase 1 of the SECC project, the construction of a sentence-level batch tool. Phase 2 will broaden the scope of both dimensions, leading to a beyond-sentence interactive tool. As to the beyond-sentence dimension, it relates to the COGRAM rules that govern the textual entities above sentence level (paragraph, subsection, section, the whole text). SECC will also handle the implementable subset of those rules, and produce appropriate error messages and possibly corrections. To build this part of the system, we are leaving the realm of the sentence-based Metal system, and enter that of text parsers. The exact approach (and its integration with the sentence-level SECC) still has to be worked out, but it is clear SGML and its associated tools will again play an important role. As to the interactive dimension, the goal is to be able to run SECC on-line on selected text fragments while working inside Interleaf. Issues here are job queue management (of batch and interactive requests to the same processor), acceptability of response times, new user interface matters (result presentation and integration).

In short, we have travelled a bit, but there is still a long way to go.

REFERENCES

1. Adriaens, G. and Schreurs D. "From Cogram to Alcogram: towards a Controlled English Grammar Checker," Proceedings of *COLING92*,595-601,1992.
2. Adriaens, G., Deprez F., De Braekeleer G., and Depoortere B. "Sisyphus: a toolkit for MT testing and evaluating," Unpublished paper, Siemens Nixdorf Software Centre Liege, 1993.

[10] For clarity's sake: both Dutch people (i.e. people living in the Netherlands) and Flemish people (i.e. people living in the northern part of Belgium) have Dutch as their language. The technical writers in the consortium are Flemish.

3. Alonso, J. A. "The Spanish Grammar Checker," *TWB (Translator's Workbench) Final Report*, 1992.

4. Humphreys, L. "The Simplified English Lexicon," *Proceedings of EURALEX92*,353-364,1992.

5. Pulman, S.G. and Rayner, M. "Computer Processable Controlled Language," *SRI International Cambridge Computer Science Research Centre*, 1994.

6. Ravin, Y. "Grammar Errors and Style Weaknesses in a Text-Critiquing System," K. Jensen, J.E. Heidorn and S.D. Richardson (eds), *Natural Language Processing: The PLNLP Approach*. Kluwer Academic Publishers, Boston, 65-76,1993.

7. Schreurs, D. and Adriaens, G. "Controlled English (CE): From COGRAM to ALCOGRAM," In P. O'Brian Holt and N. Williams (eds), *Computers and Writing, State of the Art*. Intellect Books Oxford and Kluwer Academic Publishers, Boston, 206-221,1992.

8. Steen, G.J. van der, and Dijenborg, B. "Linguistic Engineering: Tools and Products," *Proceedings of the Twente Workshop on Language Technology 2*, University of Twente, 1992.

9. Thurmair, G. "Parsing for Grammar and Style Checking," *Proceedings of COLING90*,356-370,1990.

10. Thurmair, G. "Verification of Controlled Grammars," *Esprit Translator's Workbench Report*. Also in *Translator's Workbench Final Report (abbreviated version)*, 128-132,1992.

11. Wojcik, R.H., Hoard, J.E. and Holzhauser, K.C. "The Boeing Simplified English Checker," *Proceedings of the International Conference, Human Machine Interaction and Artificial Intelligence in Aeronautics and Space*. Toulouse, Centre d'Etudes et de Recherches de Toulouse, 43-57, 1990.

12. Wojcik, R. H., Harrison, P. and Bremer, J. "Using Bracketed Parses to Evaluate a Grammar Checking Application," *Proceedings of ACL93*, 38-45, 1993.